TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Aritmetické operace v hardwaru

Milan Kolář

Ústav mechatroniky a technické informatiky

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Projekt ESF CZ.1.07/2.2.00/28.0050
Modernizace didaktických metod
a inovace výuky technických předmětů.

Aritmetické operace

Při číslicovém zpracování signálů se nejčastěji používá součet a součin – pokud používáme k popisu úroveň RTL (nebo vyšší), syntetizační nástroje použijí DSP bloky (nemusí to být vždy optimální).

Algoritmy aritmetických operací se v podstatě liší mírou paralelizace (potřebou logických buněk) a počtem hodinových taktů při *N*-bitovém datovém slovu, a s tím související spotřebou energie.

Zjednodušeně je možné je dělit na sériové, paralelní, zřetězené.

V jazyce VHDL podpora v knihovnách (numeric_std, std_logic_arith, std_logic_signed, std_logic_unsigned).

Zobrazování čísel

Čísla v hardwaru je třeba ukládat do logických obvodů (registrů, pamětí, …), příp. přenášet po sběrnicích
$\Rightarrow$ užívání dvojkové soustavy (polyadická soustava o základu z = 2):

$$A = a_{n-1}.z^{n-1} + \ldots + a_0.z^0 + a_{-1}.z^{-1} + \ldots + a_{-m}.z^{-m}$$

Polyadické soustavy zobrazují pouze nezáporná čísla
$\Rightarrow$ k zobrazování záporných čísel používáme transformace (*číselné kódy*); nejčastější:

- přímý se znaménkem
- inverzní
- doplňkový
- aditivní

Řádová mřížka

Rozdělení na celou a zlomkovou část označujeme jako *řádová mřížka,* číslo zapsané v mřížce je *slovo.*

Nelze-li zapsat číslice v nejvyšších řádech, mluvíme o *přeplnění* (přetečení, overflow), v nejnižších řádech o *ztrátě přesnosti*

n-1	n-2		0	-1			-m

n-1 … nejvyšší řád řádové mřížky (celá část má velikost n bitů)

-m … nejnižší řád řádové mřížky (zlomková část má velikost m bitů)

N … délka řádové mřížky (počet obsažených řádů) $N = n + m$

$\varepsilon = 2^{-m}$ … jednotka řádové mřížky (nejmenší zobrazitelné číslo)

$M = 2^n$ … modul řádové mřížky (nejmenší číslo, které již v řádové mřížce není zobrazitelné)

Formáty nezáporných čísel

Čísla bez znaménka (**unsigned**)

- *Obecný formát:*

$$U = u_{n-1}.2^{n-1} + \dots + u_0.2^0 + u_{-1}.2^{-1} + \dots + u_{-m}.2^{-m}$$

Rozsah zobrazitelných čísel: $0 \le U \le (2^N-1).2^{-m}$

- *Celočíselný formát* (tj. $m = 0$, $N = n$):

$$U = u_{n-1}.2^{n-1} + \dots + u_0.2^0$$

- *Zlomkový (fraction) formát* – řádová čárka se umísťuje těsně za nejvyšší bit, který je řádu 2^0 ($n = 1$, $N = m+1$)

$$U = u_0.2^0 + u_{-1}.2^{-1} + \dots + u_{-m}.2^{-m}$$

Přímý kód se znaménkem

také „přirozený kód“

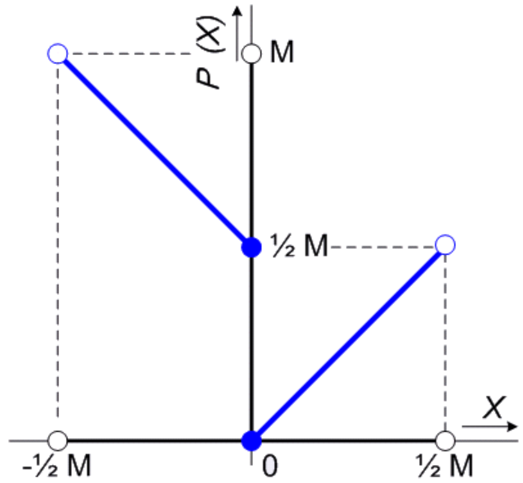
absolutní hodnota čísla se znaménkovým bitem; 0 ~ (+), 1 ~ (-)

$P(X) = X$ pro $X \geq 0$ $P(X) = |X| + ½M$ pro $X \leq 0$

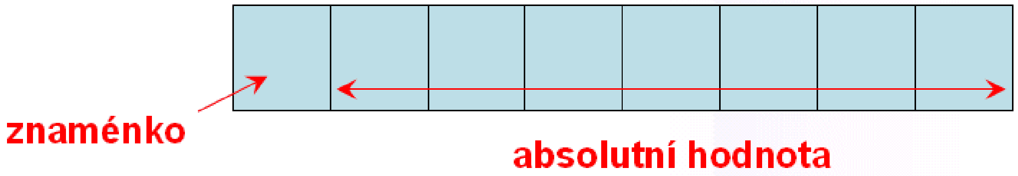
$-½M \leq X < ½M$

- složitá realizace aritmetických operací
  nejprve třeba otestovat znaménko
  pak se použije algoritmus operace
  (sčítání, odečítání)

- nevýhodou jsou dvě reprezentace nuly
  (nutno ošetřit)



Přímý kód se znaménkem

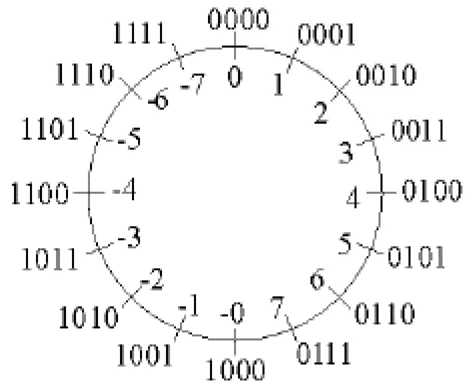


Např. pro 4-bitové slovo (n-bitové)

kladná čísla: 0 … 7 (2^{n-1}-1)

záporná čísla: -7 … 0

dvě nuly: 0000 a 1000



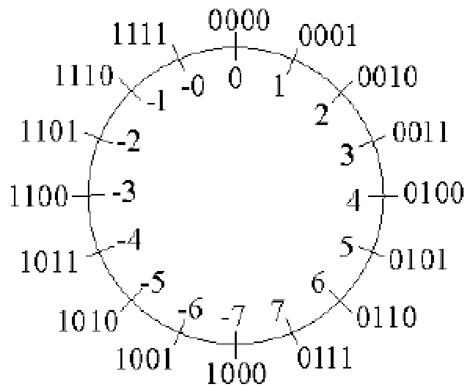
Inverzní kód

Vychází z jednotkového doplňku (one's complement)

$I(X) = X$ pro $X \geq 0$ $\qquad I(X) = M - \varepsilon + X$ pro $X \leq 0$

$-½M \leq X < ½M$

- opět problém dvou nul (0000 a 1111)
- obtížnější realizace aritmetických operací
- vznikne negací bitů daného slova
- MSB bit má opět charakter znaménka



Např. +0,8125 ~ 0,1101

–0,8125 ~ 1,0010

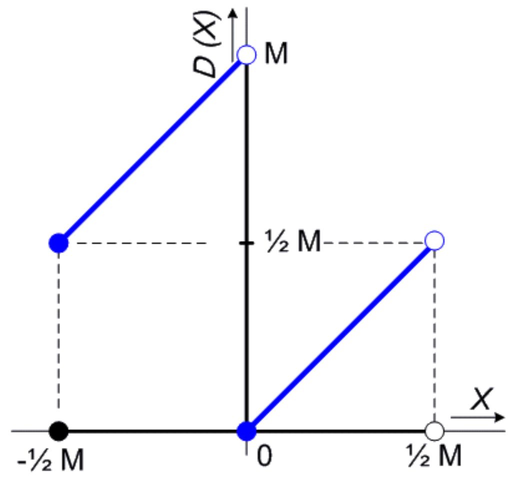
Doplňkový kód

Vychází z dvojkového doplňku“ (two’s complement)

$D(X) = X$ pro $X \geq 0$ $D(X) = M + X$ pro $X < 0$

- nejvyšší bit má opět charakter znaménka (nenese informaci o hodnotě)
- vznikne přičtením ε k inverznímu kódu
- max. záp. číslo nemá kladný ekvivalent
- algoritmus odečítání je stejný jako sčítání (sečtou se obrazy a ignoruje se přenos)
- nejpoužívanější

$-\frac{1}{2}M \leq X < \frac{1}{2}M$



Kódování čísel se znaménkem

Doplňkový kód – chápeme jako **signed** (se znaménkem):

$$D(X) = S = -s_{n-1}.2^{n-1} + \sum_{i=-m}^{n-2} s_i.2^i$$

platí pro celý rozsah *X*

$-\frac{1}{2}M \leq X < \frac{1}{2}M$

- *Celočíselný formát* (tj. $m = 0$, $N = n$):

$$S = -s_{n-1}.2^{n-1} + \sum_{i=0}^{n-2} s_i.2^i$$

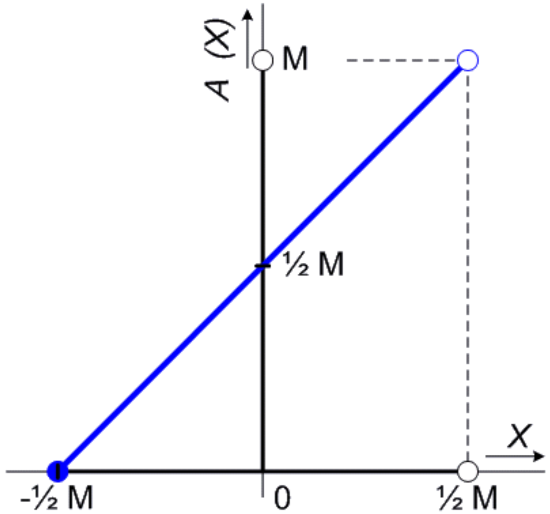
- *Zlomkový (fraction) formát* (tj. $n = 1$, $N = m+1$)

$$S = -s_0 + \sum_{i=-m}^{-1} s_i.2^i$$

Aditivní (posunutý) kód

Kód s posunutou nulou – k číslu připočteme známou konstantu K:

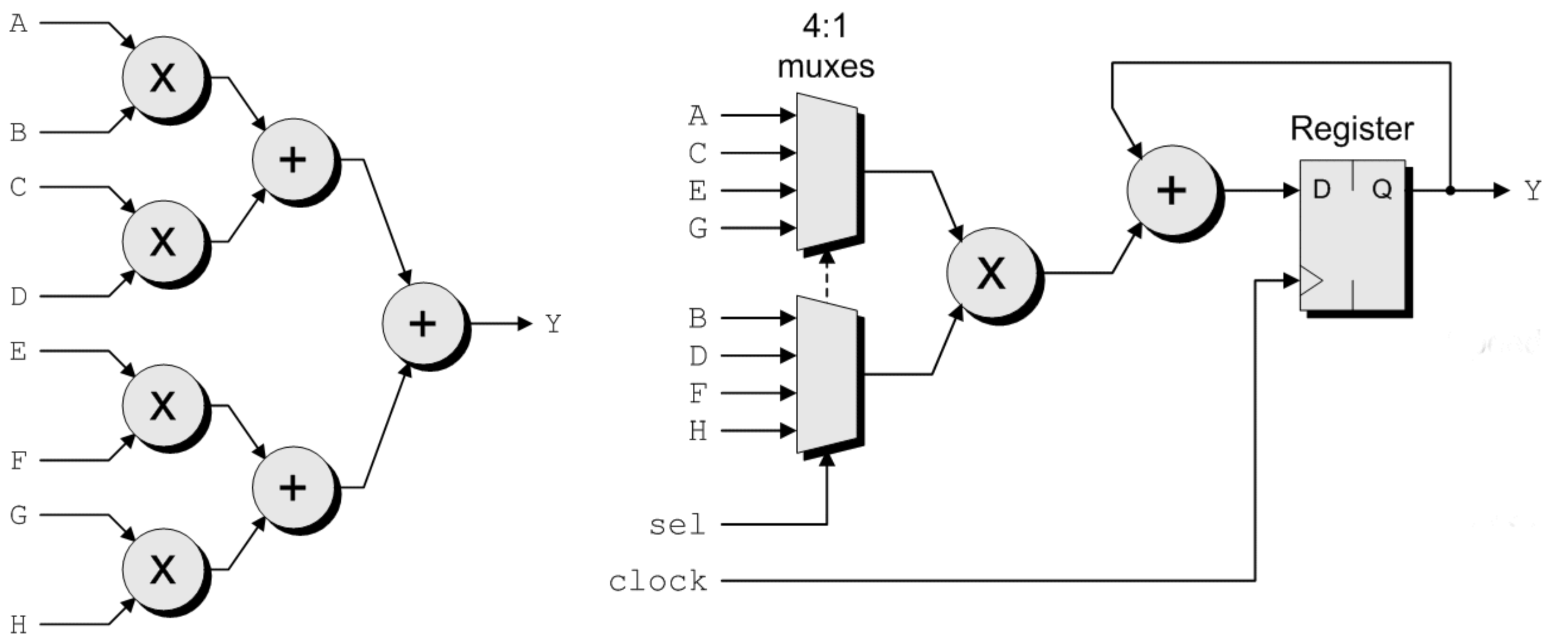
$$A(X) = X + K$$



- zachovává relace <
- především se používají 2 varianty:
  - $\Rightarrow K = 2^{n-1}$, tj. $½M$ (sudý aditivní kód)
  - $K = 2^{n-1} - \varepsilon$ (lichý aditivní kód)
- složitější realizace násobení (nejprve je třeba odečíst známou konstantu)
- převod do doplňkového kódu provedeme negací nejvyššího bitu (pro sudý aditivní kód)
- používá se pro reprezentaci exponentu reálných čísel

DSP operace

Kompromis mezi rychlostí a plochou;



někdy preferujeme rychlost i před přesností.

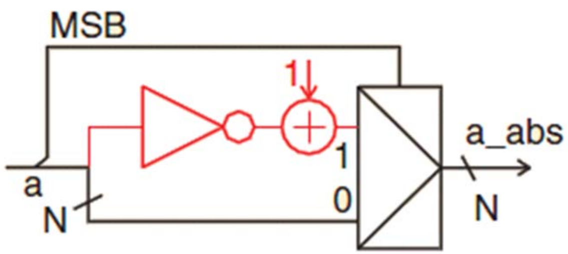
Absolutní hodnota

$D(-a) = M + (-a) = 2^n - a - \varepsilon + \varepsilon = NOT(a) + \varepsilon$

Chceme-li invertovat znaménko, invertujeme všechny bity a přičteme jedničku (u celých čísel).

ve VHDL: a_inv <= -a; nebo a_inv <= not(a) + 1;

a_abs <= unsigned(a) WHEN a(N-1) = ‘0’ ELSE unsigned(NOT(a))+1;

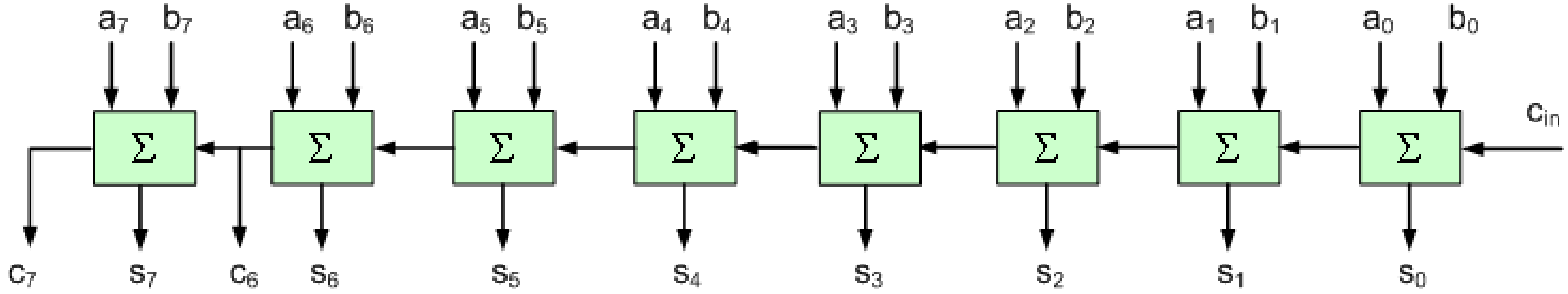


Sčítačky

Sčítačky (a odečítačky) – *poloviční* (half), *plné* (full)

Nejjednodušší je *sériová sčítačka* – úplná sčítačka s klopným obvodem pro uchování přenosu – potřeba *N* hodinových taktů.

Pokud ve VHDL zapíšeme: `c <= a + b;`
vznikne nejčastěji *paralelní sčítačka se sériovým přenosem* (ripple carry adder) – při velkém *N* raději „roztrhat“
=> *sériově-paralelní sčítačka*



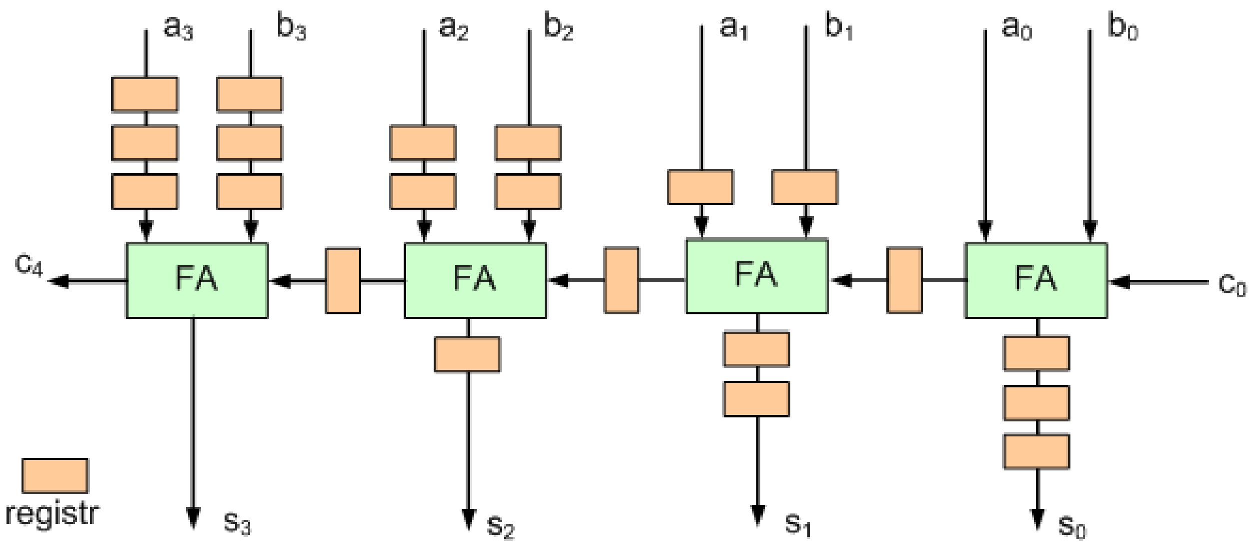
Sčítačky (pokračování)

Jazyky HDL neřeší přetečení – lépe rozšíření na N+1 bitová čísla

```
c <= (a(N-1) & a) + (b(N-1) & b);
```

Sčítačka s proudovým zpracováním – výsledek získáme za N hodinových taktů (a dále každý takt);

FA možné nahradit i vícebitovou sčítačkou

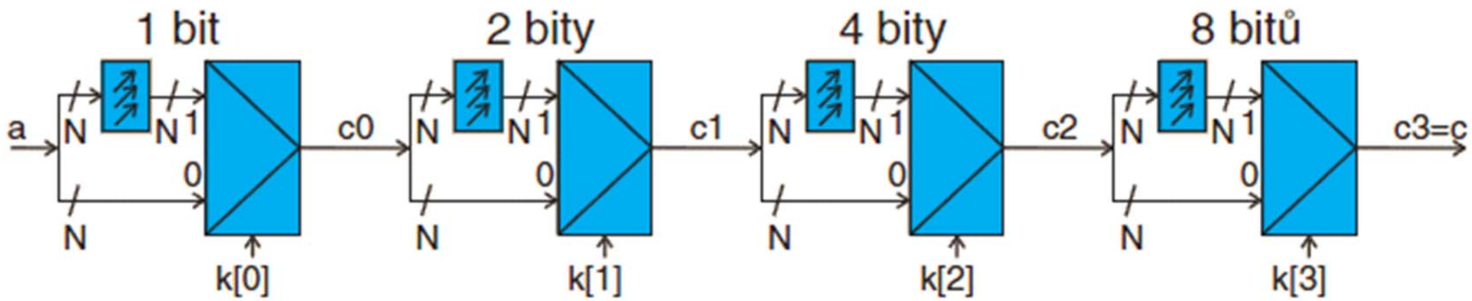


Násobičky

Nejjednodušší varianta – násobení celočíselnou mocninou dvou

=> provádí se pouze *posuv*

Barrel shifter pro posuv čísla (násobení 2^0, 2^1, …, 2^{15}).



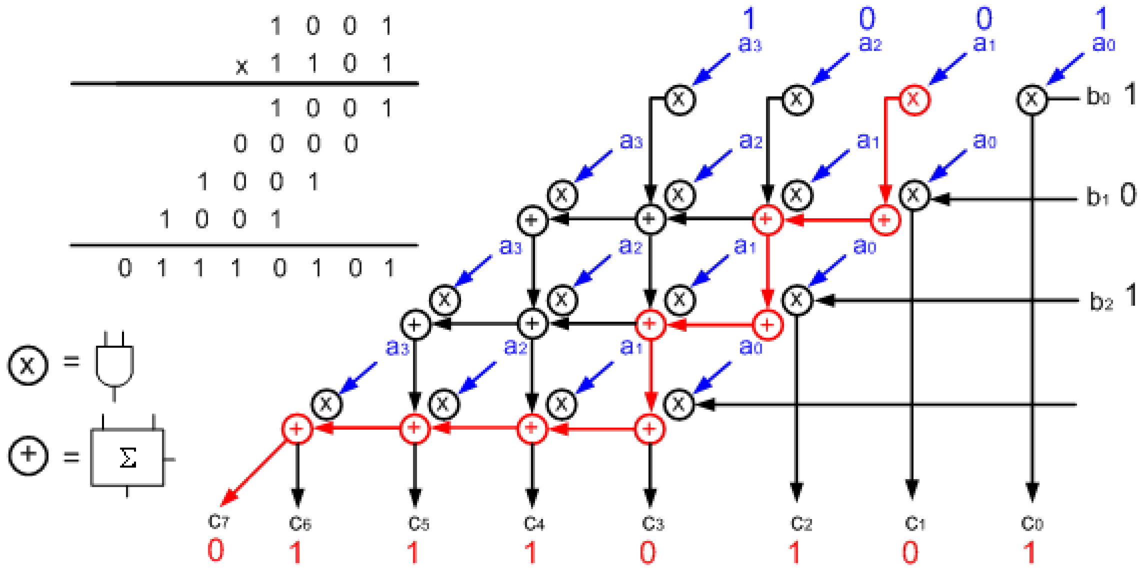
Při násobení dvou *N*-bitových čísel má výsledek 2*N* bitů.

Jednoduché řešení je *postupné sčítání* (velmi pomalé).

Násobičky (pokračování)

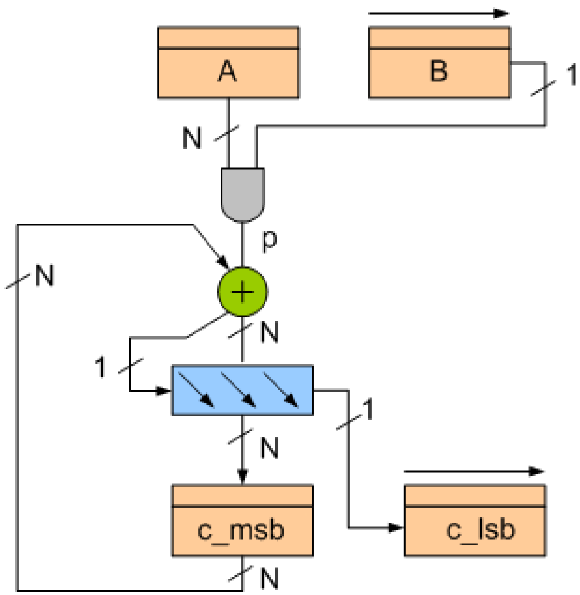
Paralelní násobička – vychází z písemného algoritmu,

- kombinační obvod z pološčítaček a log. součinů,
- rychlé, ale zabírá množství logiky,
- tato násobička vzniká při zápise ve VHDL: c <= a * b;
- pokud jsou v FPGA bloky DSP, preferují se (pro větší *N*).



Násobičky (pokračování)

Sériová násobička – jednoduchá, ale relativně pomalé (N taktů), vychází opět z písemného algoritmu.



Pro slova s velkým N používáme *sériově-paralelní varianty* nebo *násobičky s proudovým zpracováním.*

Děličky

Dělení je relativně pomalá a HW náročná operace.

Speciální případy:

Dělení celočíselnou mocninou dvou => *posuv*;
- zleva nasouváme nuly (unsigned) nebo kopii znaménka.

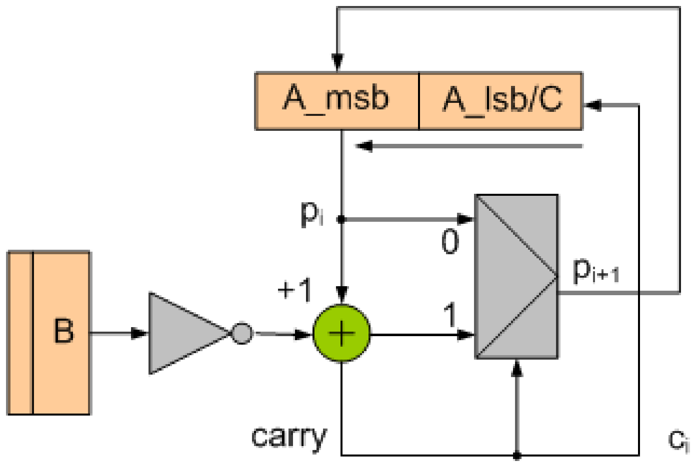
Potřebujeme-li dělit známou *konstantou*, převedeme na násobení její převrácenou hodnotou.

Nejjednodušší obecné řešení je *postupné odečítání* – pomalé.

Jinou obecnou možností je převod na *násobení převrácenou hodnotou* – problém je získání převrácené hodnoty.

Děličky (pokračování)

Sériová dělička s nezápornými operandy – vychází z algoritmu písemného dělení



Matematické knihovny ve VHDL neposkytují podporu dělení.

Druhá odmocnina

Často používaná operace, většinou počítáme celočíselně;
má-li odmocněnec 2*N* bitů, výsledek má *N* bitů.

Řada algoritmů – nejjednodušší, ale časově náročné:

Postupné hledání: for y = 1 to A
if A – y*y ≤ 0 then return(y-1);
end;

Metoda bisekce (půlení intervalu) – počáteční odhad výsledku, spočítáme mocninu, porovnáme, zvolíme spodní nebo horní polovinu původního intervalu.

Rekurentní algoritmy:

$Y_0=X$, $Y_{i+1}=0{,}5.(X/Y_i+Y_i)$, kde X je vstupní hodnota

Goniometrické funkce

Použití v algoritmech čísl. zpracování signálu, v generátorech průběhů, v počítačové grafice aj.

Záleží, zda potřebujeme generovat spojitý průběh nebo libovolnou hodnotu (bez předešlé historie).

- Použití tabulky (paměť ROM) – stačí 1 kvadrant, náročné na plochu, rozlišení lze zlepšit lineární interpolací;
- Použití mocninných řad (Taylorův rozvoj aj.);
- Metoda CORDIC – iterační metoda založená na rotaci vektoru okolo počátku o předem známé úhly.

Logaritmus, exponenciální funkce

Méně časté, občas logaritmické transformace operandů před výpočtem
=> násobení a dělení převádí na sčítání a odečítání, obecnou mocninu a odmocninu na násobení a dělení.

Algoritmy: na principu aproximace,
rekurentní (konvergentní řady).