Aritmetické operace v digitálních obvodech

BPC-NDI 2022

Autor: Vojtěch Dvořák

16. 11. 2022

Osnova

- Operace sčítání a odčítání
  - Jednobitová sčítačka
  - Vícebitová sčítačka
  - Reprezentace znaménkových čísel
- Operace násobení
- Reprezentace čísel ve formátu pevné řádové čárky (Fixed-Point)

Operace sčítání

- Kombinační logika
- Součet realizován pomocí řady jednobitových sčítaček
- Šířka součtu je šířka největšího z operandů + 1 (přetečení do vyššího řádu)
- Stejná struktura pro znaménková i bezznaménková čísla

- Příklad sčítání: $14_d\,(1110_b) + 11_d\,(1011_b)$

Jednobitová sčítačka

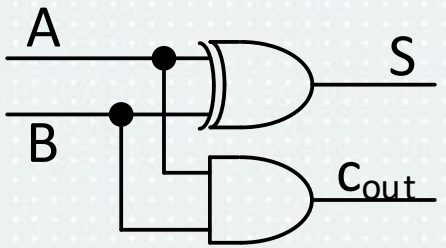
- Neúplná sčítačka
  - Dva (jednobitové) vstupy *A* a *B* a dva (jednobitové) výstupy *S* (součet) a c_{out} (přenos do vyššího)
  - Logické funkce

$$S = A \oplus B$$

$$c_{out} = AB$$

A	B	c_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

  - Pro vytvoření vícebitové sčítačky je přenos z nižšího řádu připojen na vstup A nebo B → potřeba dvou neúplných sčítaček pro (každý) bit



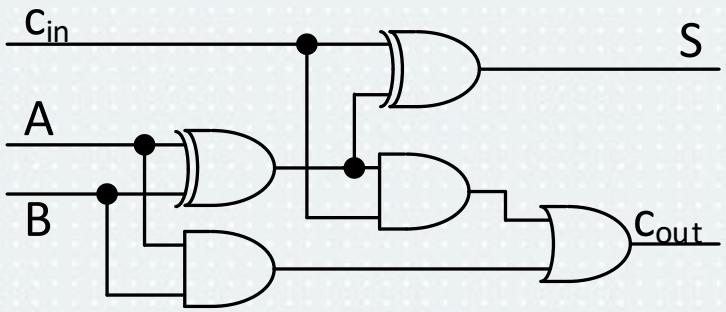
Jednobitová sčítačka

- Úplná sčítačka
  - Tři (jednobitové) vstupy A, B a c_{in} a dva (jednobitové) výstupy S (součet) a c_{out} (přenos do vyššího)
  - Logické funkce

$$S = A \oplus B \oplus c_{in}$$

$$c_{out} = AB + c_{in}(A \oplus B)$$

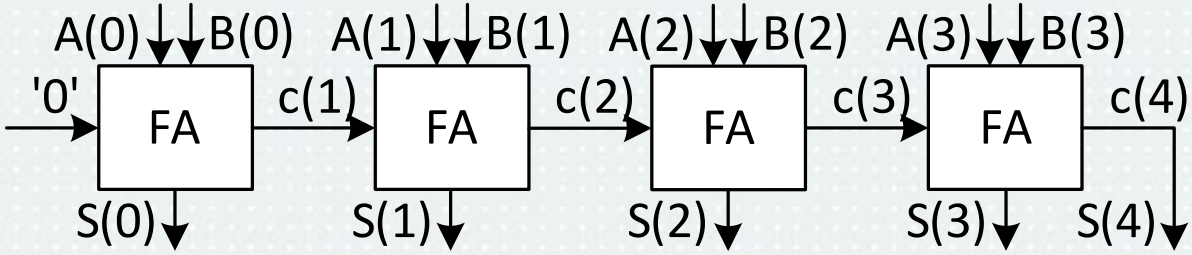
  - Efektivně dvojice neúplných sčítaček



A	B	c_{in}	c_{out}	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

Vícebitová sčítačka

- Kaskádní zapojení jednobitový úplných sčítaček
- Sčítačka s postupným přenosem (carry-ripple adder)
  - Pomalá* (dlouhá kombinační cesta), ale efektivní z hlediska spotřeby zdrojů



- Sčítačka s paralelním přenosem (carry-lookahed adder)
  - Rychlá (kratší kombinační cesta), ale náročná na spotřebu zdrojů v FPGA

** FPGA jsou obvykle optimalizovány pro tuto strukturu*

Vícebitová sčítačka

- Sčítačka s postupným přenosem je použita ve sloze numeric_std

```
function ADD_UNSIGNED ( L,R: UNSIGNED; C: STD_LOGIC ) return UNSIGNED  is
constant L_left:INTEGER:= L'length-1;
alias XL: UNSIGNED(L_left downto 0) is L;
alias XR: UNSIGNED(L_left downto 0) is R;
variable RESULT: UNSIGNED(L_left downto 0);
variable CBIT : STD_LOGIC:= C;
begin
  for i in 0 to L_left loop
    RESULT(i) := CBIT xor XL(i) xor XR(i);
    CBIT := (CBIT and XL(i)) or (CBIT and XR(i)) or (XL(i) and XR(i));
    end loop;
  return RESULT;
  end ADD_UNSIGNED;
```

Modulární aritmetika

- „Hodinová aritmetika“
- Čísla v definovaném rozsahu (*modulo*)

$$N_M \cong N \bmod M$$

- „Přetečení“ je modulární redukce
- Všechny aritmetické operace v digitálních obvodech jsou přirozeně prováděny v modulární aritmetice kvůli konečnému počtu bitů
- Např. čítač
  - Čítání → 7 → 0 → … → 7 → 0 →
- Příklad sčítání: 14_d (1110_b) + 11_d (1011_b)

→ Volba modula závisí na velikosti výsledku

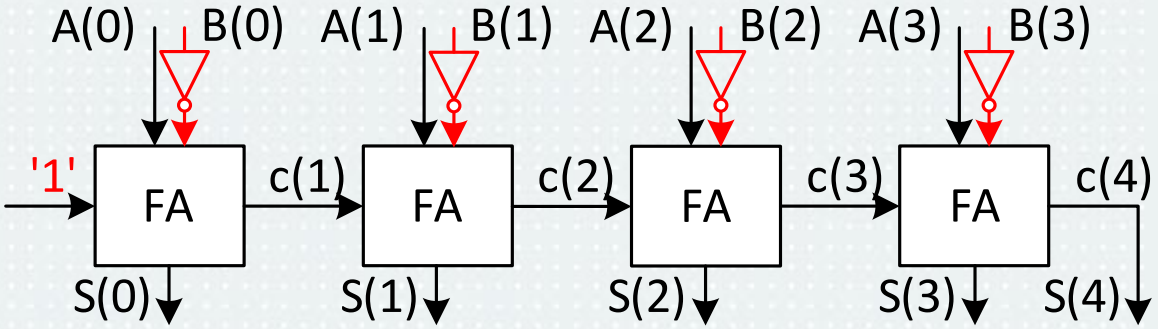
bin	*dec*
011	3
100	4
101	5
110	6
111	7
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7
000	0
001	1
010	2

Záporná čísla

- Dvojkový doplněk

$$A_{neg} = \text{not}(A) + 1$$

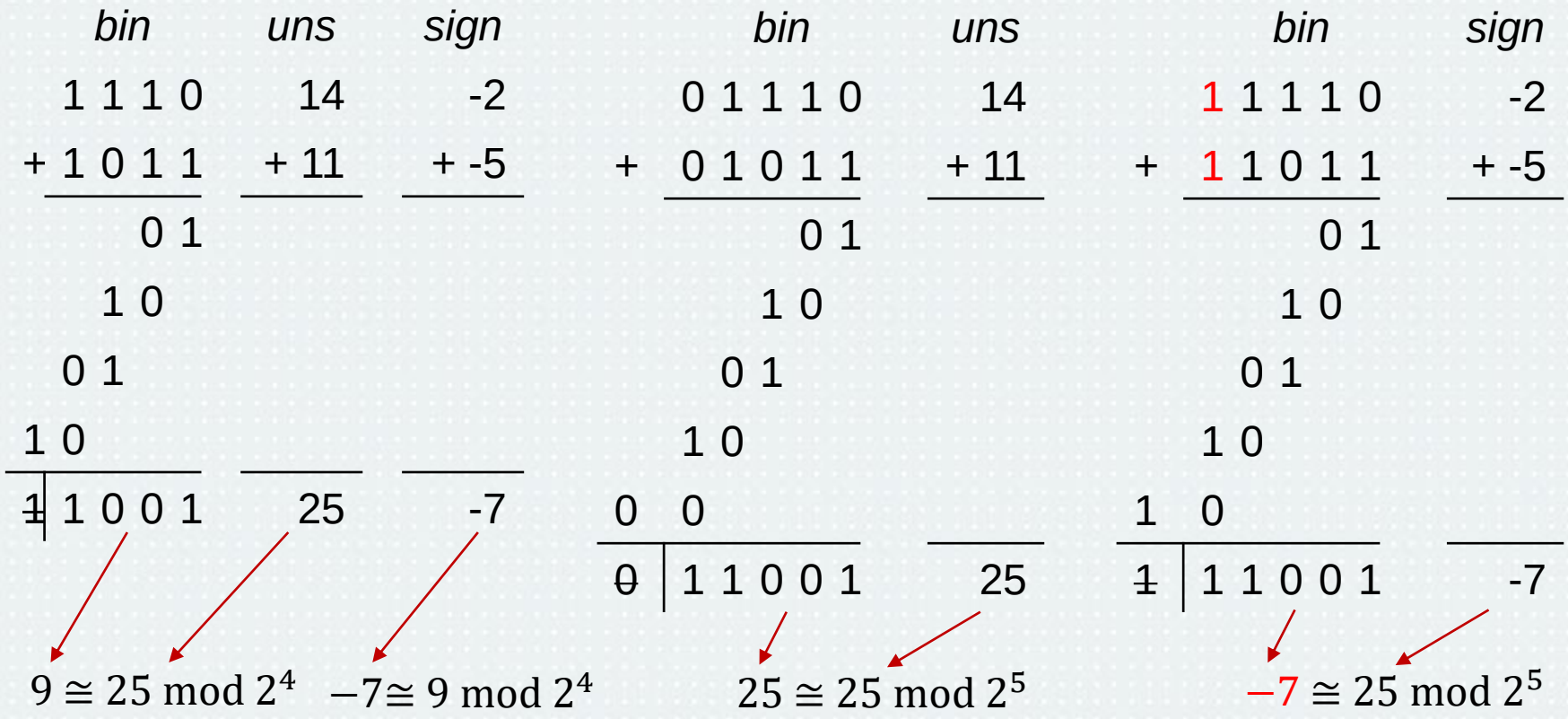
- Odčítání je sčítání se záporným číslem → převedení druhého operandu do dvojkového doplňku
- „Odčítačka“ je modifikace sčítačky:



- Operátor “-” `return ADD_UNSIGNED (L01,not(R01),'1');`
- Příklad sčítání: $-2_d\ (1110_b) + -5_d\ (1011_b)$

bin	*uns*	*sign*
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1
000	0	0
001	1	1
010	2	2

Modulární aritmetika



Sloha numeric_std

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
...
signal a_uns, b_uns : unsigned (3 downto 0);
signal s_uns, d_uns : unsigned (4 downto 0);
signal a_sgn, b_sgn : signed   (3 downto 0);
signal s_sgn, d_sgn : signed   (4 downto 0);
...
s_uns <= resize(a_uns, 5) + b_uns; -- add unsigned
d_uns <= resize(a_uns, 5) - b_uns; -- subtract unsigned
s_sgn <= resize(a_sgn, 5) + b_sgn; -- add signed
d_sgn <= resize(a_sgn, 5) - b_sgn; -- subtract signed
```

Funkce *resize* rozšíří číslo o (další) znaménkové bity podle datového typu
Pokud je argument menší než vstupní číslo, dojde k odstranění (dalších) znaménkových bitů

https://www.csee.umbc.edu/portal/help/VHDL/packages/numeric_std.vhd

Operace násobení

- Kombinační logika
- Jednobitová násobička – logický součin
- Šířka součinu je součet šířek obou operandů
- Přenos do vyšších řádů realizován pomocí řady jednobitových sčítaček
- Počet sčítaček ~ N^2
- Násobení znaménkových čísel je odlišné
  - Převod obou hodnot na kladná čísla → násobení kladných čísel → konverze výsledku dle znamének
- Redukce zpoždění - Wallace Tree Multiplier

	bin	*dec*
	1 1 1 0	14
	x 1 0 1 1	x 11
	0 0 0 0 1 1 1 0	014
	0 0 0 1 1 1 0 0	140
	0 0 0 0 0 0 0 0	
	0 1 1 1 0 0 0 0	
	1 0 0 1 1 0 1 0	154

Sloha numeric_std

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
...
signal a_uns, b_uns : unsigned (3 downto 0);
signal p_uns        : unsigned ((2*4)-1 downto 0);
signal a_sgn, b_sgn : signed   (3 downto 0);
signal p_sgn        : signed   ((2*4)-1 downto 0);
 ...
a_uns <= a_uns * b_uns; -- mult unsigned
b_sgn <= a_sgn * b_sgn; -- mult signed
```

Žádné rozšíření vstupních čísel není třeba, operátor “*“ vrací hodnotu o šířce součtu šířek operandů. Signál (proměnná), do něhož je výsledek násobení přiřazen, musí být správné šířky.

https://www.csee.umbc.edu/portal/help/VHDL/packages/numeric_std.vhd

Shrnutí – sčítání, odčítání, násobení

- Operace sčítání a odčítání se znaménkovými i bezznaménkovými čísly jsou ekvivalentní, záleží jen na způsobu reprezentace (*chápání*) čísel.
  Operace násobení je odlišná pro znaménková a bezznaménková čísla.

  → Ve VHDL pomocí datových typů *signed* a *unsigned*
- Všechny aritmetické operace v digitálních obvodech přirozeně zahrnují modulární aritmetiku – potřeba rozšířit číslo tak, aby nedošlo k modulární redukci.

  → Ve VHDL pomocí funkce *resize*
- Operace sčítání (operátor +), odčítání (operátor -) a násobení (operátor *) jsou definovány pro oba datové typy v knihovně numeric_std, používejte jen tyto funkce.

Reprezentace čísel v digitálních obvodech

- Celá čísla $\boldsymbol{Z}$
  - Viz. předchozí slajdy
- Racionální čísla $\boldsymbol{Q} = \frac{a}{b}; a, b \in Z$
  - Pevná řádová čárka

$$\boldsymbol{Q_{FX}} = \frac{a}{2^{c_w}}; a, c_w \in Z, c_w = konst.$$

  - Plovoucí řádová čárka

$$\boldsymbol{Q_{FP}} = \frac{a}{2^{p_w}}; a, p_w \in Z, p_w \neq konst.$$

- Reálná čísla nelze přesně reprezentovat (nekonečný počet číslic/bitů)

Reprezentace čísel v pevné řádové čárce

- Řádová čárka ~ desetinná čárka, její pozice se nemění
  $a = N * 2^{-W}$; *N je celé číslo a W je pozice řádové čárky*
- Části čísla
  - Celá část (C) – rozsah hodnot $[0; 2^{C} - 2^{-Z}]$
  - Zlomková část (Z) – rozlišení hodnot = 2^{-Z}

N(7)	N(6)	N(5)	N(4)	N(3)	N(2)	N(1)	N(0)
C(3)	C(2)	C(1)	C(0)	Z(3)	Z(2)	Z(1)	Z(0)

řádová čárka

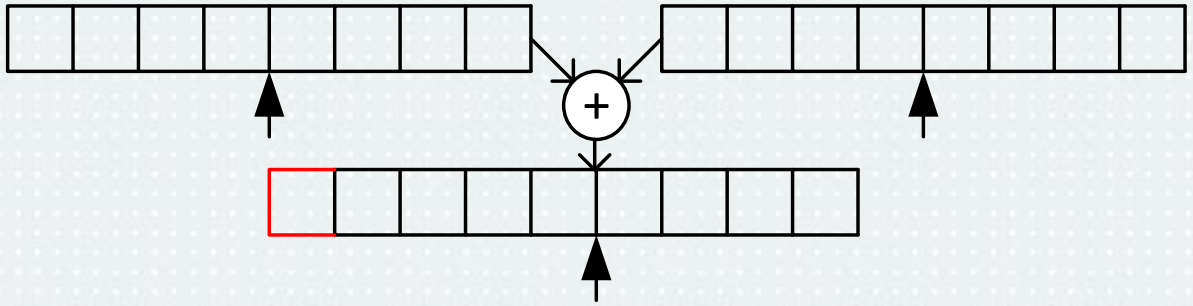
- Příklad – N(8, 4) = 123
  - Maximální hodnota: $2^{4} - 2^{-4} = 15{,}9375$
  - Rozlišení hodnot: $2^{-4} = 0.0625$
  - Hodnota v *dec*: $a = N * 2^{-W} = 123 * 2^{-4} = 7{,}6875$

Operace s čísly v pevné řádové čárce

- Operace sčítání

$$a_1 = N_1 * 2^{-W};\ a_2 = N_2 * 2^{-W}:$$
$$a_1 + a_2 = N_1 * 2^{-W} + N_2 * 2^{-W}$$
$$= (N_1 + N_2) * 2^{-W}$$

  - Operace sčítání nemění pozici řádové čárky
  - Stále je třeba počítat s rozšířením čísel, aby nedošlo k neošetřenému přetečení (modulární redukci)



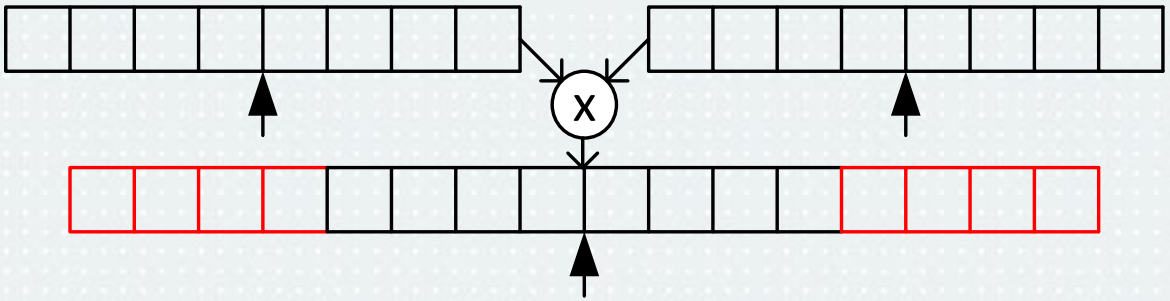
Operace s čísly v pevné řádové čárce

- Operace násobení

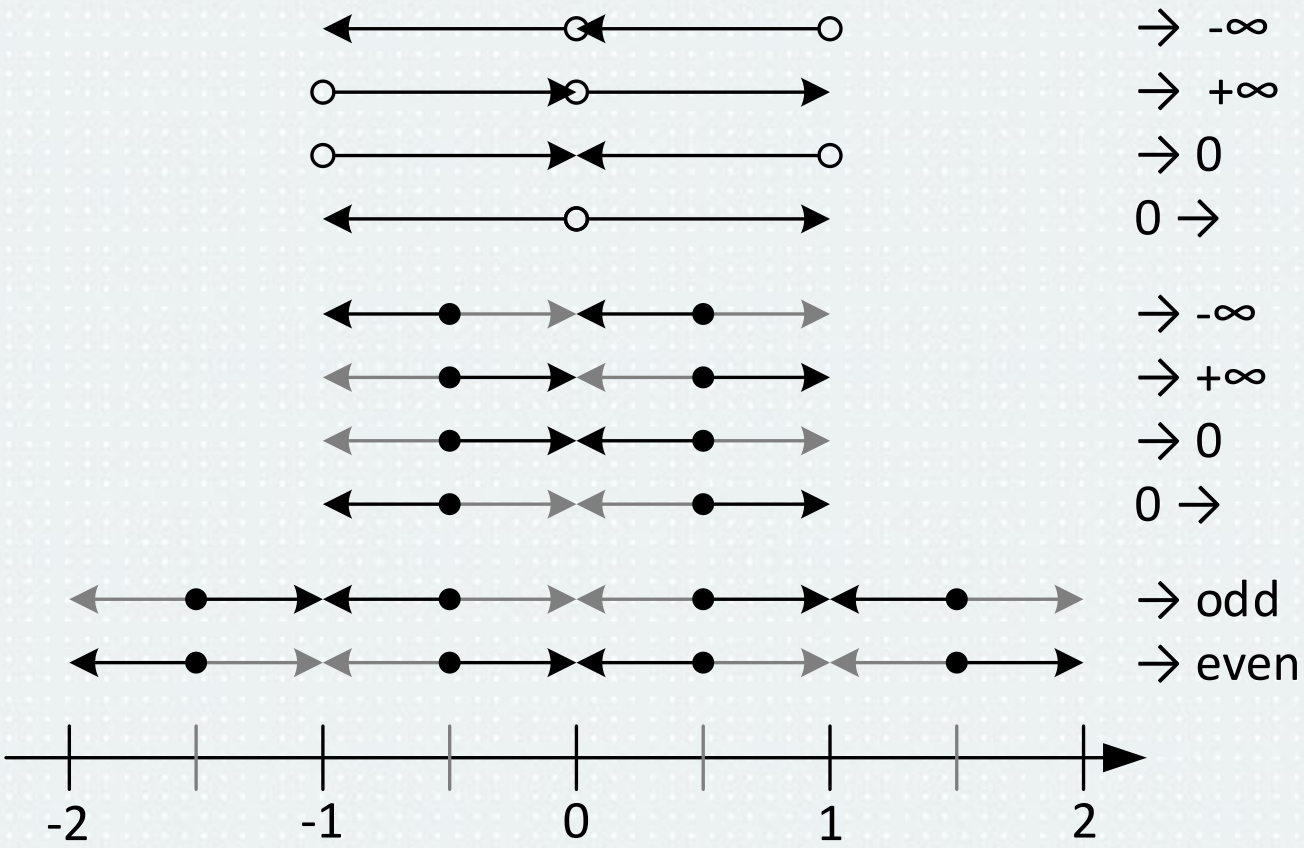
$$a_1 = N_1 * 2^{-W};\ a_2 = N_2 * 2^{-W}:$$
$$a_1 + a_2 = N_1 * 2^{-W} * N_2 * 2^{-W}$$
$$= (N_1 * N_2) * (2^{-W} * 2^{-W})$$
$$= (N_1 * N_2) * (2^{-2*W})$$

  - Operace násobení **mění** pozici řádové čárky $(2^{-W} \rightarrow 2^{-2*W})$ a rozšiřuje číslo $(N_1, N_2 \rightarrow N_1 * N_2)$

→ pro převod zpět do stejné reprezentace je třeba „vybrat“ správné bity



Způsoby zaokrouhlení čísel



- Metoda zaokrouhlení *floor* (zaokrouhlení k -∞) pouze odstranění spodních bitů

děkuji za pozornost

(a pište komentáře)