

# Układy arytmetyczne. Układy iteracyjne. Liczby zmiennoprzecinkowe.

Data wykonania 11.12.2020

**Zadanie 2.d** Sumator wielobitowy można zrealizować jako układ iteracyjny zbudowany z łańcucha połączonych sumatorów 1-bitowych. Czy na podobnej zasadzie można zrealizować układ realizujący odejmowanie? Jak będą wyglądały funkcje różnica i pożyczka?

2d

$a_i, b_i, c_i$	Odejm	Pożyczka
0 0 0	0	0
0 0 1	1	1
0 1 0	1	1
0 1 1	0	1
1 0 0	1	0
1 0 1	0	0
1 1 0	0	0
1 1 1	1	1

Odejm		$c_i$
$a_i, b_i$	0	1
0 0	0	1
0 1	1	0
1 1	0	1
1 0	1	0

Pożyczka		$c_i$
$a_i, b_i$	0	1
0 0	0	1
0 1	1	1
1 1	0	1
1 0	0	0

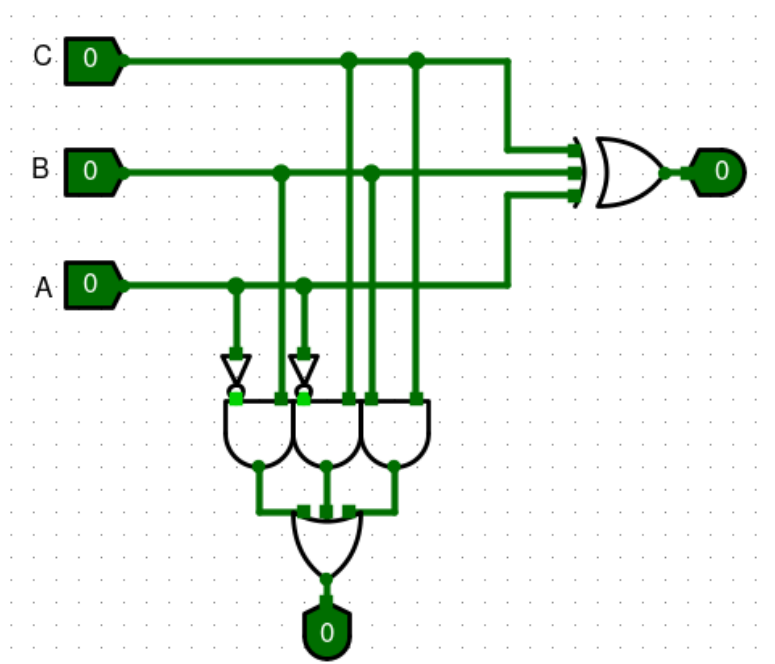
  

$$\text{Odejm} = \bar{a}b\bar{c} + \bar{a}b\bar{c} + abc + a\bar{b}\bar{c}$$

$$\text{Pożyczka} = \bar{a}c + \bar{a}b + bc$$
  

$$\text{Odejm} = \text{XOR}(a, b, c)$$

Zdjęcie 1: Wyprowadzenie funkcji różnica i pożyczka.



Zrzut ekranu 1: Schemat układu odejmującego w Logisim



Sprawdzenia wyniku dokonano za pomocą strony internetowej:  
<https://www.exploringbinary.com/floating-point-converter/>

## Decimal

Enter a decimal number (e.g., 3.1415, 1.56e-11, 4e20) (no suffixes, commas, operators)

8.4

Convert

Clear

### Options:

- Precision (check one or both): ☐ Double ☒ Single
- Output formats (check all desired):
  - ☐ Decimal (e.g., 122.75)
  - ☒ Binary (e.g., 1111010.11)
  - ☐ Normalized decimal scientific notation (e.g.,  $1.2275 \times 10^2$ )
  - ☒ Normalized binary scientific notation (e.g.,  $1.11101011 \times 2^6$ )
  - ☐ Normalized decimal times a power of two (e.g.,  $1.91796875 \times 2^6$ )
  - ☐ Decimal integer times a power of two (e.g.,  $491 \times 2^{-2}$ )
  - ☐ Decimal integer times a power of ten (e.g.,  $12275 \times 10^{-2}$ )
  - ☐ Hexadecimal floating-point constant (e.g., 0x1.ebp6)
  - ☒ Raw binary (e.g., 0 10000101 111010110000000000000000)
  - ☒ Raw hexadecimal (e.g., 42f58000)

## Floating-Point

Converts to this binary floating-point number (selected forms shown):

### Binary

Single:

1000.0110011001100110011

### Normalized Binary Scientific Notation

Single:

1.0000110011001100110011  $\times 2^3$

### Raw Binary (sign field | exponent field | significand field)

Single:

0 10000010 0000110011001100110

### Raw Hexadecimal (sign field|exponent field|significand field)

Single:

41066666

Zrzut ekranu 3: Sprawdzenie obliczeń wykonanych w zadaniu 9.