# Działania arytmetyczne

– sposób wykonania
    dokładność obliczeń
    postać wyniku
– poprawność wyniku - konieczna sygnalizacja
– formowanie wyniku

arytmetyka stałoprzecinkowa (pozycyjna/uzupełnieniowa dwójkowa)
– nieograniczony zakres
– nieograniczona dokładność

arytmetyka zmiennoprzecinkowa (działania na rekordach/polach kodu)
– ograniczony zakres, trudne rozszerzanie
– ograniczona dokładność

arytmetyka nasyceniowa (wektorowa z dyskryminacją)
– ustalony zakres, rozszerzanie niemożliwe
– ustalona dokładność

## Arytmetyka - problemy

arytmetyka stałoprzecinkowa (pozycyjna/uzupełnieniowa dwójkowa)

– nadmiar w dodawaniu lub odejmowaniu – konieczna sygnalizacja
  rozwiązanie – rozszerzenie znakowe/zerowe

– mnożenie – zwykłe, dokładne (podwójnej długości)
  – dolne – wykrywanie nadmiaru
  – górne – błąd zaokrąglenia

– dzielenie – wynik: iloraz i reszta

arytmetyka zmiennoprzecinkowa (działania na rekordach/polach kodu)

– wyjątki
– normalizacja
– zaokrąglenia
– brak łączności i przemienności dodawania

arytmetyka nasyceniowa (wektorowa z dyskryminacją)

– sposób dyskryminacji
– ograniczony repertuar: dodawanie, odejmowanie i mnożenie?

## Dodawanie i odejmowanie

$$x_i \pm y_i \pm c_i = \pm 2c_{i+1} + s_i \Rightarrow x_i \pm y_i = s_i \pm (2c_{i+1} - c_i)$$

kod naturalny NB ($m$ – rozmiar operandu/długość słowa)

$$X = \sum_{i=0}^{m-1} x_i 2^i, \quad Y = \sum_{i=0}^{m-1} y_i 2^i \quad \Rightarrow \quad S = X \pm Y = \sum_{i=0}^{m-1} s_i 2^i \pm c_m 2^m$$

IA-32 - rozkaz **add /adc** oraz **sub /sbb**
- flaga przeniesienia **CF**

kod uzupełnieniowy U2 ($m$ – rozmiar operandu/długość słowa)

$$X = -x_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} x_i 2^i \qquad Y = -y_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i$$

$$\Downarrow$$

$$S = X \pm Y = -s_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} s_i 2^i \mp (c_m - c_{m-1}) 2^m$$

IA-32 - rozkaz **add /adc** oraz **sub /sbb**
- flaga nadmiaru stałoprzecinkowego **OF**

- odejmowanie – dodanie liczby przeciwnej: $X - Y = X + (-Y)$
- dodawanie – odejmowanie liczby przeciwnej: $X + Y = X - (-Y)$

## Kod liczby przeciwnej

kod liczby przeciwnej – uzupełnienie do zera:

$$-X = 0 - X = -1 + 1 - X$$
$$\downarrow$$
$$-X = (-1 - X) + 1 = [-2^{m-1} + (2^{m-2} + \ldots 2^1 + 2^0) - X] + 1$$
$$\downarrow$$
$$-X = [(-2^{m-1} + \sum_{i=0}^{m-2} 2^i) - (-x_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} x_i 2^i)] + 1$$
$$\downarrow$$
$$-X = [-(1 - x_{m-1}) 2^{m-1} + \sum_{i=0}^{m-2} (1 - x_i) 2^i] + 1 = \overline{X} + 1$$

algorytmy mnemotechniczne:

- zaneguj wszystkie bity oryginału i do uzyskanego kodu dodaj pozycyjnie „1"
- zaneguj wszystkie bity oryginału,
  **oprócz** prawostronnego ciągu zer i poprzedzającej go „1"
  (propagacja dodanej „1" kończy się na pozycji najniższej „1" oryginału)

IA-32 - rozkaz **neg** – nierozszerzalny! (ogólnie: algorytm odejmowania od 0)
- flaga nadmiaru stałoprzecinkowego **OF**

# Dodawanie i odejmowanie liczb naturalnych jako rozszerzeń U2

$$\left|\{x_{m-1},\ldots,x_1,x_0\}_{\mathrm{NB}}\right| = \left|\{0,x_{m-1},\ldots,x_1,x_0\}_{\mathrm{U2}}\right|$$

$$\downarrow$$

($s_m = c_m$), ponadto w dodawaniu ($c_{m+1}=0$), w odejmowaniu ($c_{m+1}=c_m$)

$$\downarrow$$

$$S = -(0\pm 0)2^m + \sum_{i=0}^{m-1}(x_i \pm y_i)2^i = \sum_{i=0}^{m-1} s_i 2^i \pm c_m 2^m$$

Odejmowanie liczb naturalnych przez dodanie uzupełnienia

$$-\left|\{x_{m-1},\ldots,x_1,x_0\}_{\mathrm{NB}}\right| = \left|\{1,(1-x_{m-1}),\ldots,(1-x_1),(1-x_0)\}_{\mathrm{U2}}\right| + 1$$

$$\downarrow$$

$$x_i + (1-y_i) + c_i = 2c_{i+1} + s_i$$

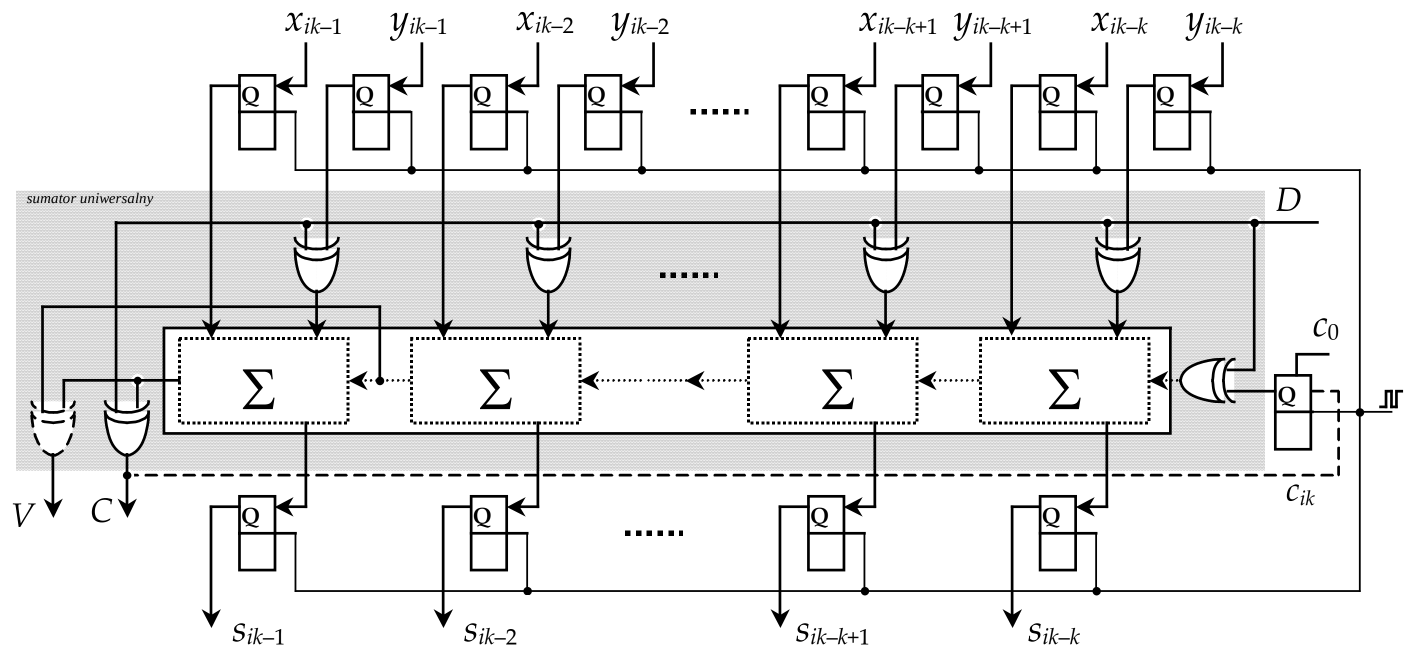
$$\downarrow$$

$$S = -(0+1)2^m + \sum_{i=0}^{m-1}(x_i + (1-y_i))2^i + 1 = \sum_{i=0}^{m-1} s_i 2^i - (1-c_m)2^m$$

# Rozszerzenie zakresu dodawania/odejmowania

odejmowanie – dodanie uzupełnienia (także w systemie naturalnym)

**wniosek**: można skonstruować sumator uniwersalny




długie dodawanie/odejmowanie: powiązanie kolejnych słów – bit przeniesienia

**wskaźnik poprawności** (końcowa sygnalizacja przekroczenia zakresu):

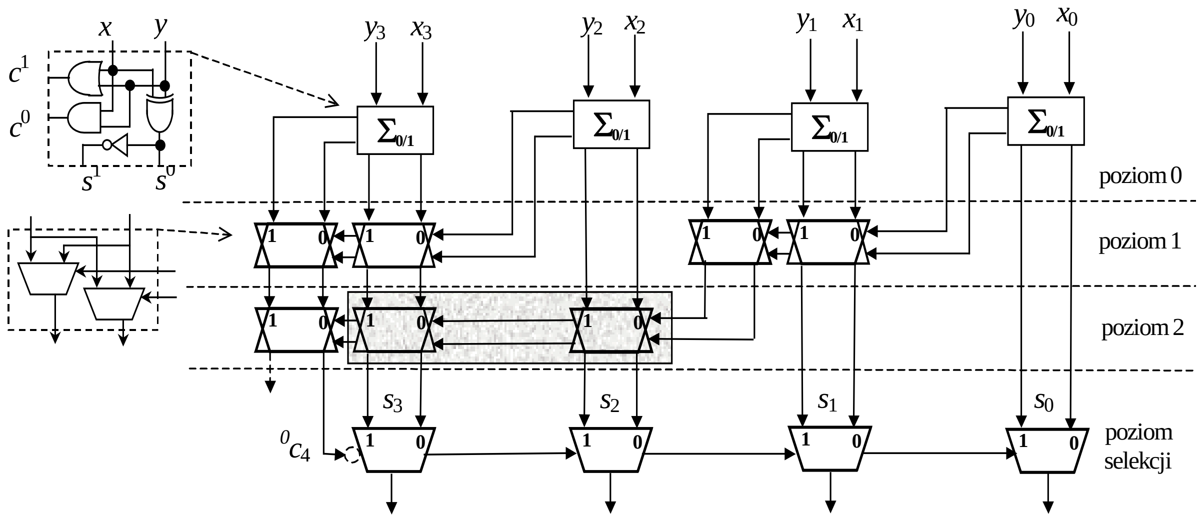
- C – dla kodu naturalnego
- V – dla kodu uzupełnieniowego

## Szybkość dodawania

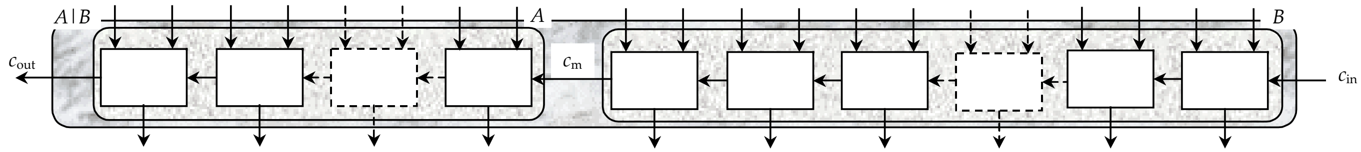
- czas dodawania zależy od *szybkości propagacji przeniesień*

rozwiązania:

- szybki układ wytwarzania przeniesień – CLA, PPA
- tworzenie alternatywnych sum dla grup bitów – COSA, CSLA



# Propagacja i generowanie przeniesień – analiza




$c_{out}$=1 jeśli:

- $c_{in}$=1 jest przesyłane przez blok *AB* do wyjścia $c_{out}$:
  - o $c_{in}$=1 jest przesyłane przez blok *B* do wyjścia $c_m$, a następnie przez blok A do wyjścia
- w bloku AB jest tworzone $c_{out}$=1, zaś $c_{in}$ jest dowolne
  - o w bloku A jest wytwarzane $c_{out}$=1, zaś $c_m$ jest dowolne
  - o w bloku B jest wytwarzane $c_m$=1, a następnie przez blok A przesyłane do wyjścia $c_{out}$

$$c_{out} = G_A + P_A G_B + P_B P_A\, c_{in/B} = G_{BA} + P_{BA}\, c_{in/B}$$

Stąd przeniesienie na *i*-tą pozycję sumatora jest równe:

$$c_i = G_{0,i-1} + P_{0,i-1} c_0$$

– schemat wyznaczania funkcji $G_{0,i}$ i $P_{0,i}$ można optymalizować

– wszystkie funkcje $G_{0,i}$ i $P_{0,i}$ można obliczyć w czasie O($\lceil \log_2 n \rceil$)

## Sumatory prefiksowe (PPA)

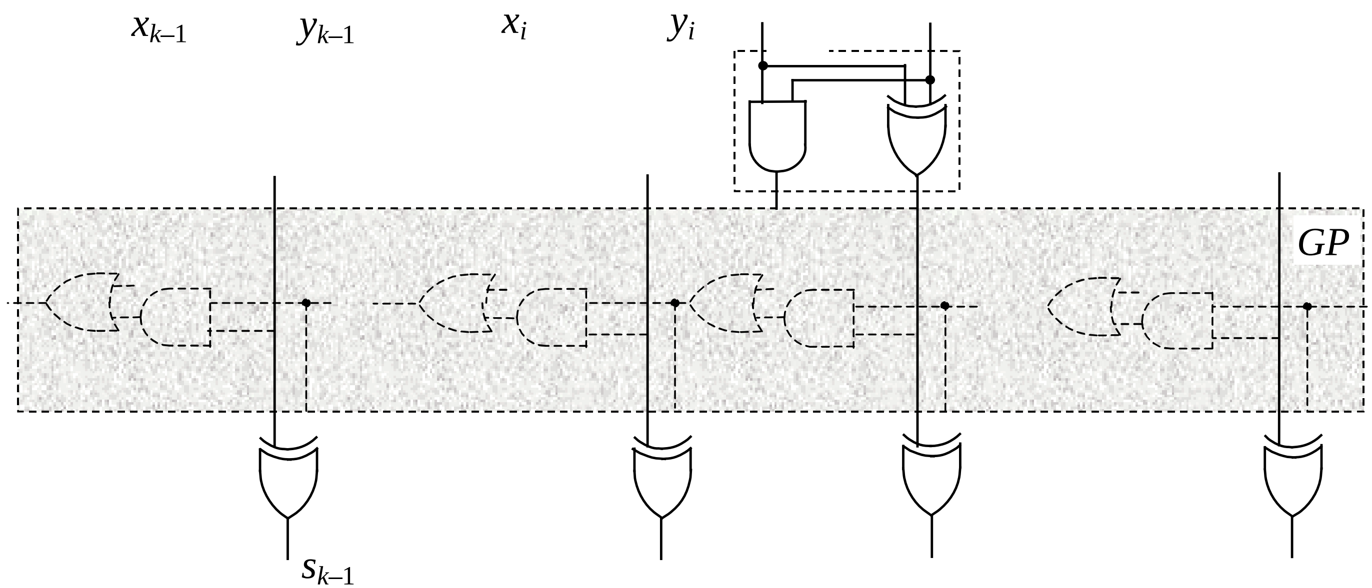
sumator prefiksowy – *parallel prefix adder*, PPA ($k \geq s \geq i$):

$$G_{i,k} = G_{s+1,k} + P_{s+1,k} G_{i,s},$$

$$P_{i,k} = P_{i,s} P_{s+1,k}.$$

$$c_{k+1} = G_{i,k} + P_{i,k} c_i$$

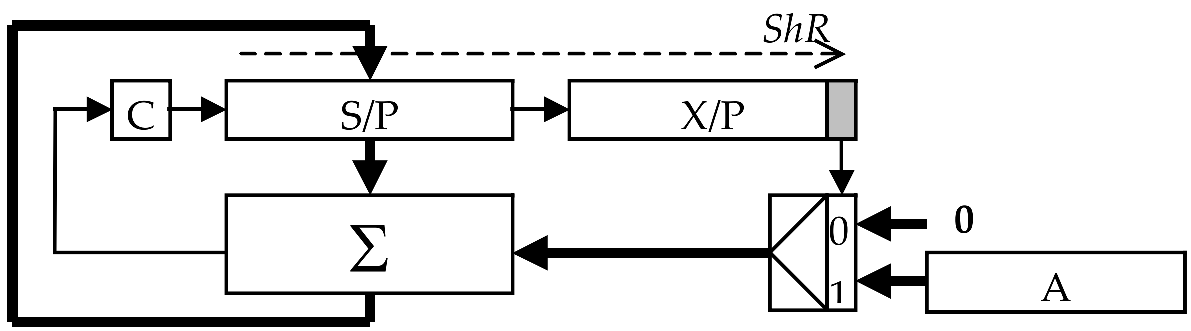
$$s_i = h_i \oplus c_i = (\text{gdy } c_0 = 0) \; h_i \oplus G_{0,i-1}$$




Blok GP – wytwarzanie wartości przeniesień $c_i = G_{0,i-1}$

# Mnożenie dwójkowe

$$XA = \sum_{i=0}^{i=k-1} Ax_i 2^i$$




S/P – rejestr sumy częściowej i iloczynu górnego, X/P – rejestr mnożnika i iloczynu dolnego, C – przeniesienie, A – rejestr mnożnej, ShR – przesunięcie o jedną pozycję w prawo

szybkość akumulacji iloczynów częściowych (n – liczba bitów mnożnika:

– dodawanie sekwencyjne – czas liniowy $O(n)$

– dodawanie wieloargumentowe – czas logarytmiczny $O(\log n)$

# Przyśpieszenie mnożenia – reduktor CSA

dodawanie wieloargumentowe – idea:

zamiana $k$ argumentów o tej samej wadze (na danej pozycji)

na $m$ argumentów o różnych wagach (zapis pozycyjny)

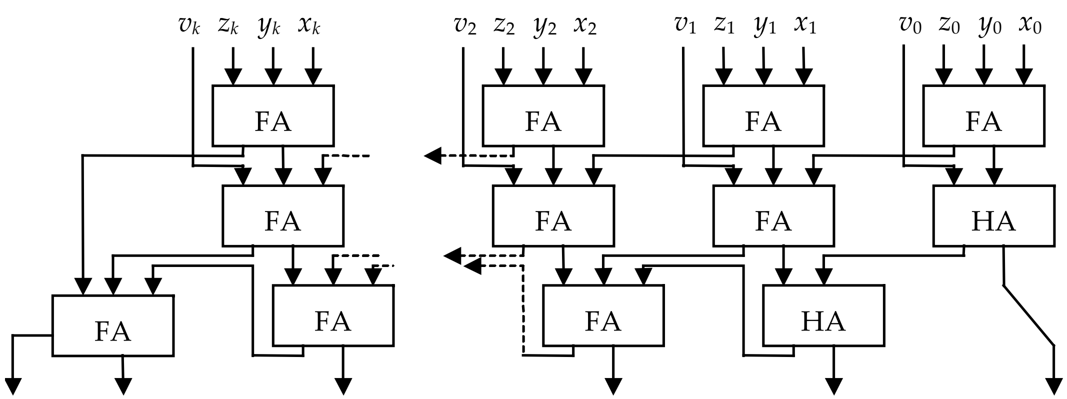
$$\beta^i(x_i^{(1)}+x_i^{(2)}+\ldots+x_i^{(k)})=\beta^i(u_i^{(0)}+u_i^{(1)}\beta^1+\ldots++u_i^{(m-1)}\beta^{m-1})$$

$$m=\left\lceil \log_\beta[k(\beta-1)+1]\right\rceil$$

realizacja: (system dwójkowy: $k$=3, $m$=2)

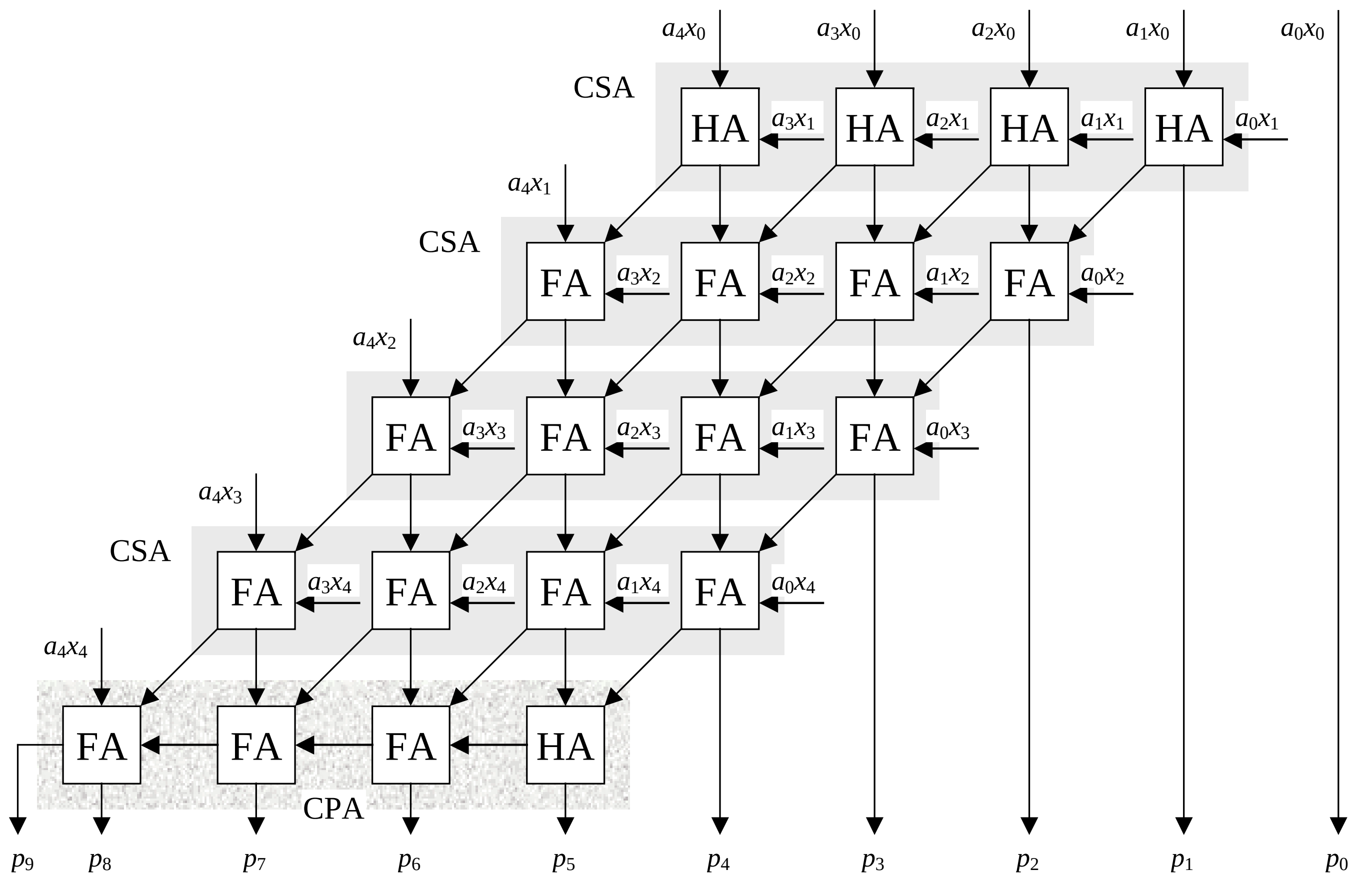
– matryca mnożąca – struktura liniowa, redukcja $k$-$m$ argumentów na 1 poziom

– drzewo Wallace'a – $n$ argumentów: liczba poziomów $2(\sqrt{2})^l \le n \le 2(\frac{3}{2})^l$



# Szybkie mnożenie – matryca

## Matryca: redukcja sekwencyjna




Matryca mnożąca – liniowa struktura CSA (argumenty: iloczyny elementarne $a_i x_j$)

## Mnożenie w dwójkowym systemie uzupełnieniowym

- uwzględnianie rozszerzeń – dodatkowe bity
- zamiana na argumenty dodatnie i korekcja iloczynu

$$B_{U2} = -b_{m-1}2^{m-1} + \sum_{j=0}^{j=m-2} b_j 2^j = -2^{m-1} + (1-b_{m-1})2^{m-1} + \sum_{j=0}^{j=m-2} b_j 2^j = -2^{m-1} + B_{NB}^{*}$$

$$X_{U2}A_{U2} = \sum_{i=0}^{k-2} x_i 2^i A_{U2} + x_{k-1}(-A_{U2})2^{k-1}$$

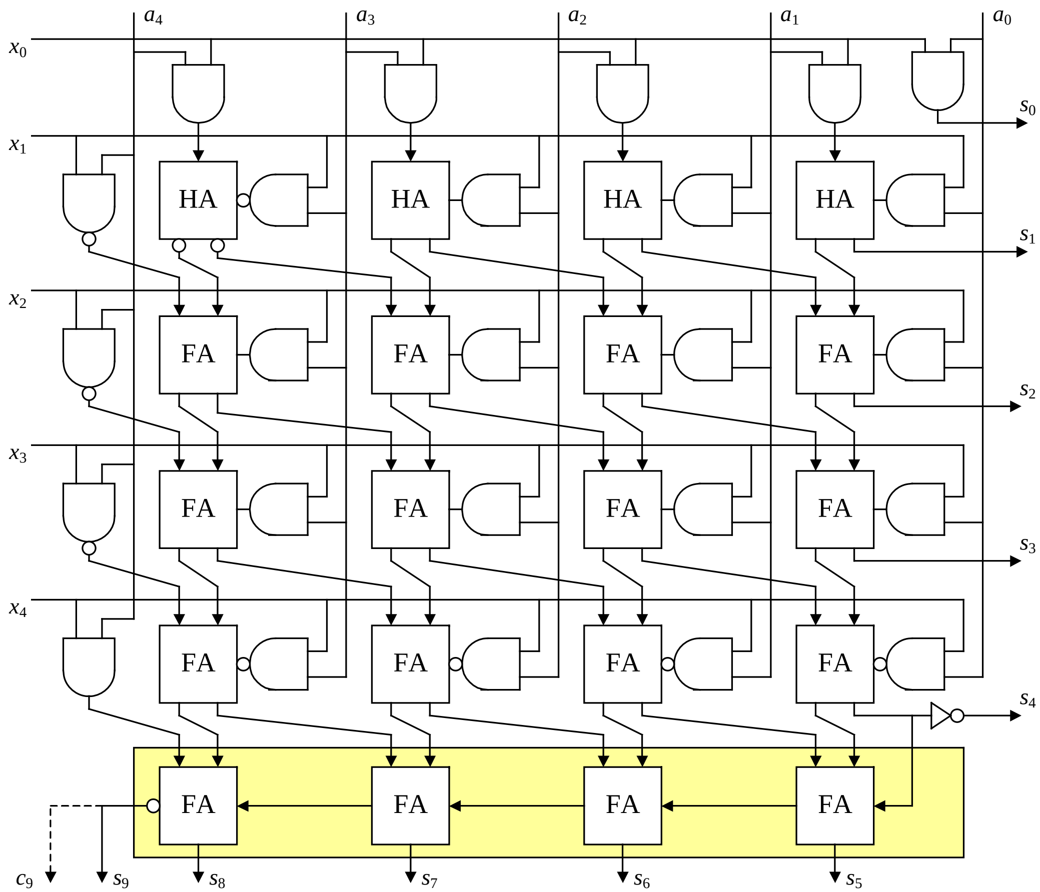
$$x_i A_{U2} = -x_i a_{m-1} 2^{m-1} + \sum_{j=0}^{j=m-2} x_i a_j 2^j =$$

$$= -2^{m-1} + (1 - x_i a_{m-1})2^{m-1} + \sum_{j=0}^{j=m-2} (x_i a_j)2^j = -2^{m-1} + Ax_{NB}^{(i)}$$

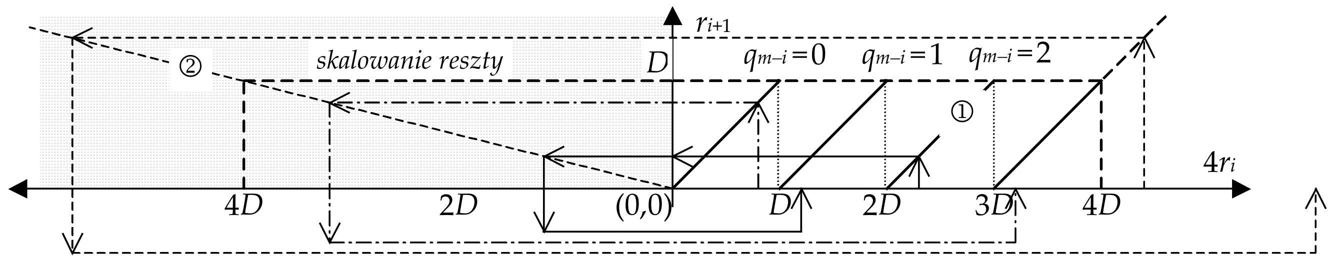
Stąd wynika, że

$$X_{U2}A_{U2} = \sum_{i=0}^{k-1} 2^i A_{NB}^{(i)} + 2^{m-1} - 2^{k+m-1}$$

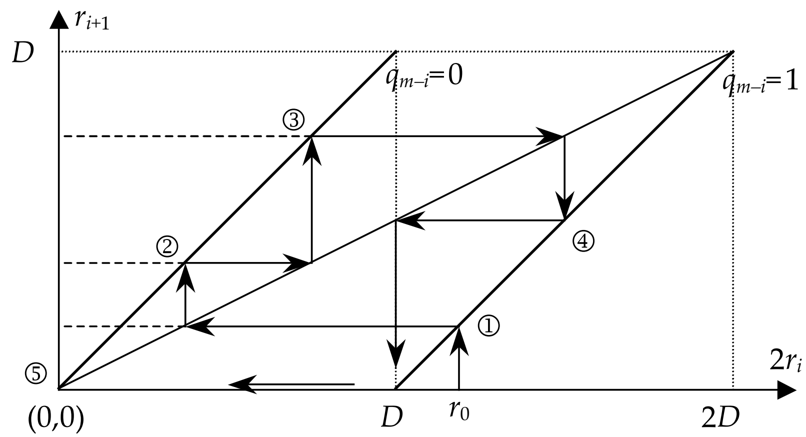
# Szybkie mnożenie w kodzie uzupełnieniowym



# Dzielenie dwójkowe




Wykres dzielenia: $r_{i+1} = \beta r_i - q_{m-i}D, \quad 0 \le r_{i+1} < D$ (podstawa $\beta$=4)



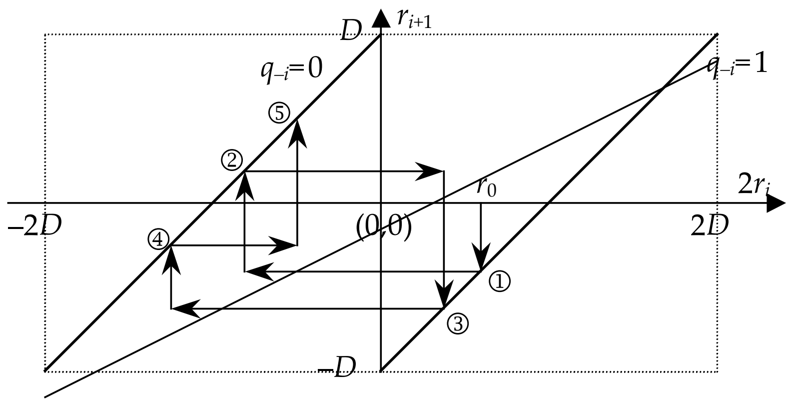

Dzielenie w systemie dwójkowym – kolejne bity ilorazu: 10010

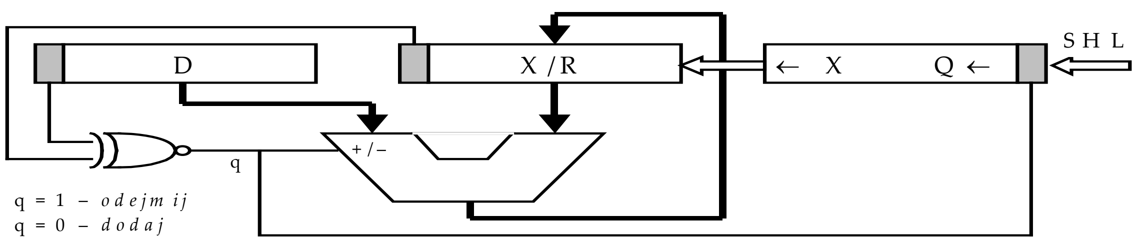
# Dzielenie nieodtwarzające

→ normalizacja ilorazu ($|D/2|<|2^{-m}X|<|D|$)

$XD<0 \rightarrow r_0=2^{-m}X+D,\ q_0=\underline{1}=(1);\ \ XD>0 \rightarrow r_0=2^{-m}X-D,\ q_0=0=(0);$

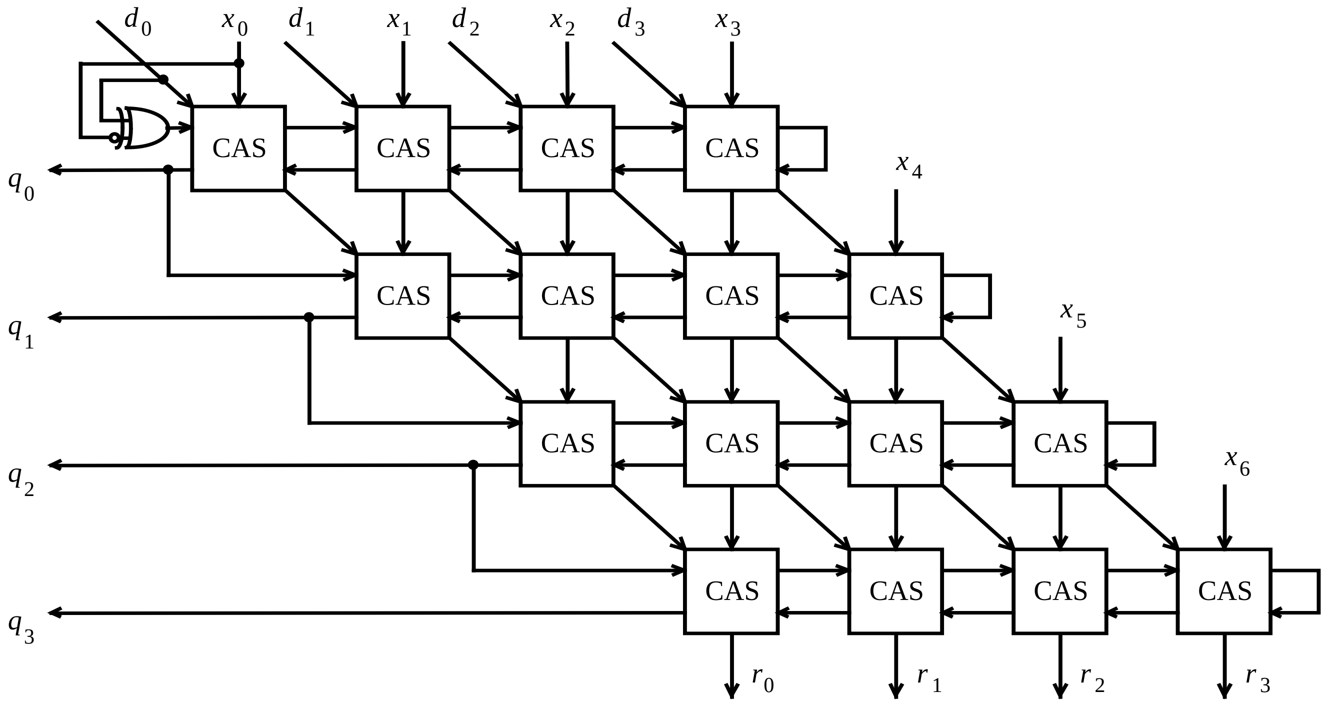
$$q_{-i}=\begin{cases}0, \text{ gdy } r_i\cdot D<0,\\ 1, \text{ gdy } r_i\cdot D\geq 0,\end{cases}\qquad r_{i+1}=2r_i+(1-2q_{-i})D$$




X/R – rejestr reszt, Q – rejestr ilorazu, D – rejestr dzielnika

# Dzielenie nieodtwarzające w matrycy




- odejmowanie/dodawanie z propagacją przeniesień skrośnych
- czas dzielenia w matrycy zawierającej $n$ wierszy jest rzędu $n^2$

# Działania zmiennoprzecinkowe

*Formaty zmiennoprzecinkowe IEEE 754-2008*

| Parametr | Symbol | 32b | 64b | 128b | extended |
|---|---|---|---|---|---|
| Rozmiar formatu | $n$ | 32 | 64 | 128 | $32t$ ($t \geq 8$) |
| Rozmiar mnożnika* | $m$ | 23 (+ 1) | 52 (+1) | 112 (+1) | $32\,t - e$ |
| Rozmiar wykładnika | $e$ | 8 | 11 | 15 | $\geq 16$ |
| Obciążenie wykładnika | $N$ | 127 | 1023 | 16383 | $\geq 32767$ |
| Zakres wykładnika | $E$ | [−126,+127] | [−1022,+1023] | [−16382,+....] | $[-(N-1),+N]$ |
| Dokładność * | *ulp* | $2^{-23} \approx 10^{-7}$ | $2^{-52} \approx 10^{-15}$ | $2^{-112} \approx 10^{-31}$ | $2^{-m+1}$ |
| Zakres formatu | *RNG* | $\cong 2^{128}$ $\cong 3{,}8 \cdot 10^{38}$ | $\cong 2^{1024}$ $\cong 9 \cdot 10^{307}$ | $\geq 2^{16384}$ | |

*Problemy*

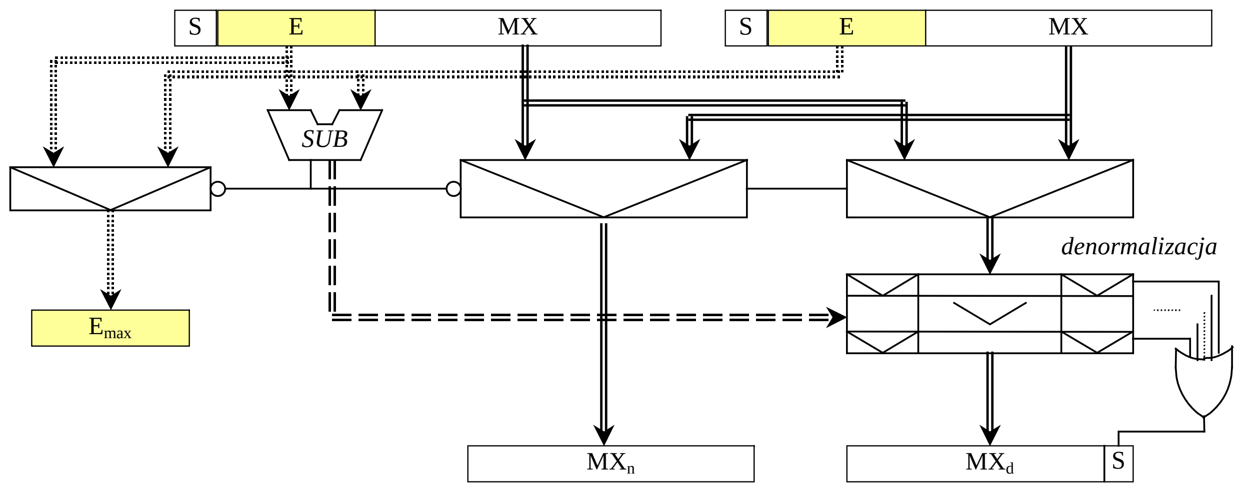
- niedokładność
- zaokrąglanie i normalizacja
- nadmiar lub niedomiar $\Rightarrow$ obsługa (skalowanie wyniku)
- nie-liczby

# Wyrównanie argumentów zmiennoprzecinkowych

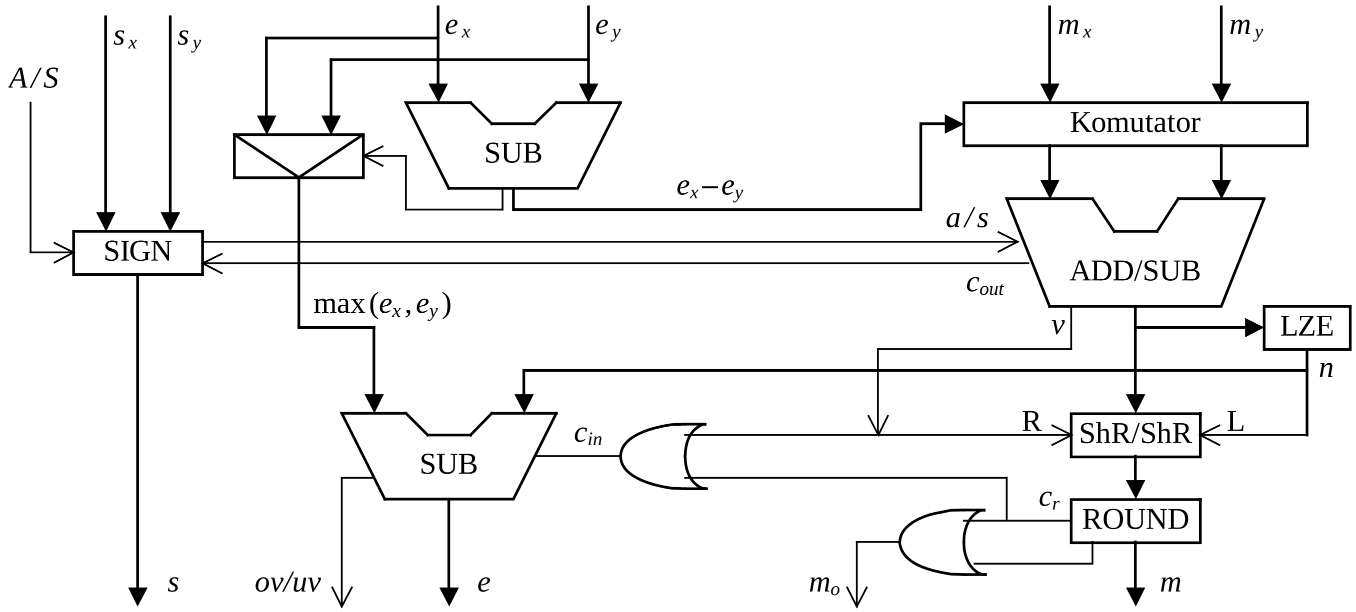
Dodawanie i odejmowanie

$$F_1 \pm F_2 = M_1 \beta^{E_1} \pm M_2 \beta^{E_2} = \beta^{E_1}(M_1 \pm M_2 \beta^{E_2 - E_1}) \qquad (E_1 > E_2)$$

zwykle konieczne wyrównanie – denormalizacja argumentu



# Sumator zmiennoprzecinkowy

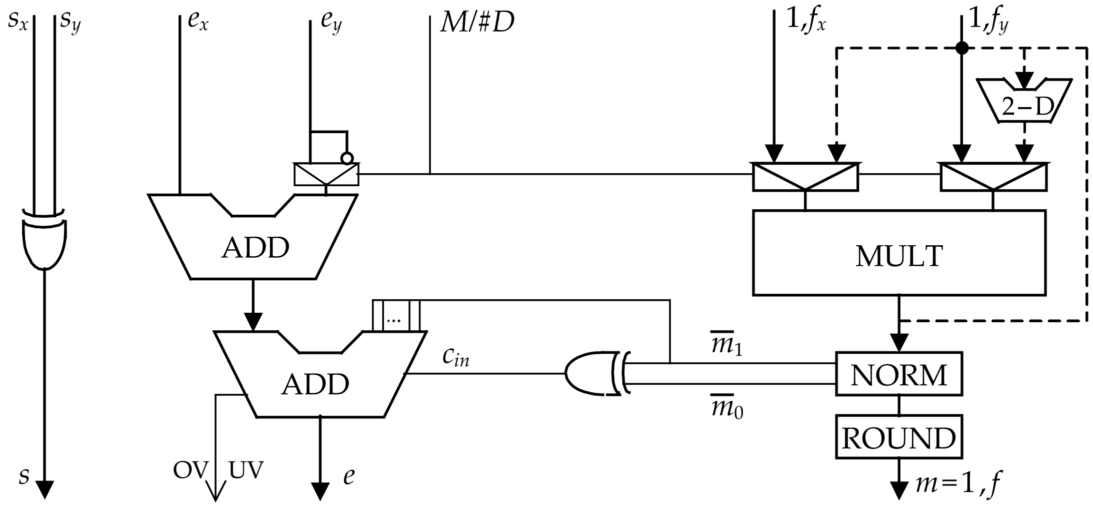



*Moduł wykładnika*: SIGN – generator znaku wyniku, MPX – multiplekser wyboru wykładnika wyniku, SUB – układ odejmujący wykładniki, ALIGN – sterowanie denormalizacją znaczników. *Moduł znacznika*: ADD/SUB – sumator znaczników, ShR – układ przesunięcia w prawo, LZE – koder wiodących zer, ShR/ShL – układ postnormalizacji, ROUND – układ zaokrąglania.

# Zmiennoprzecinkowy układ mnożąco-dzielący

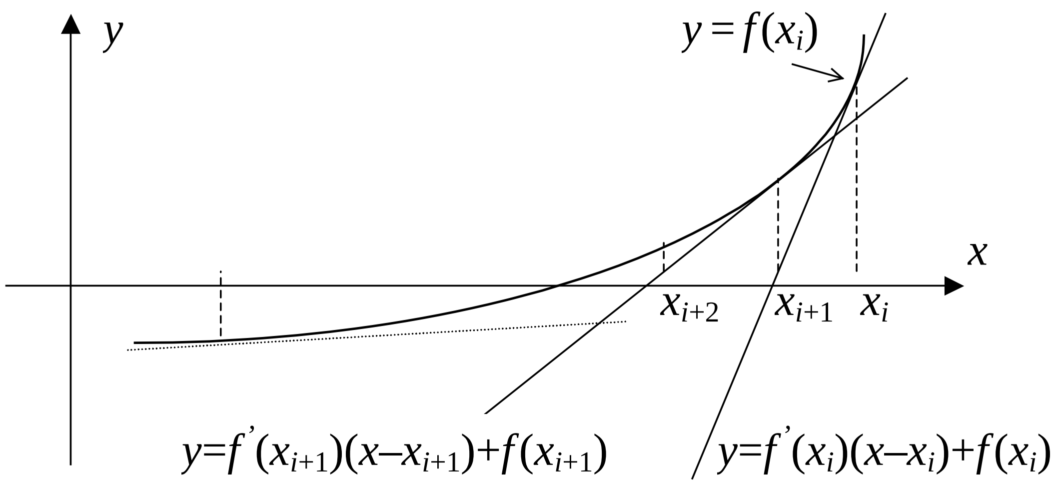
Mnożenie i dzielenie (# – mnożenie lub dzielenie)

$$F_1 \# F_2 = M_1\beta^{E_1} \# M_2\beta^{E_2} = (M_1 \# M_2)\beta^{E_1 \pm E_2}$$




Mnożenie zmiennoprzecinkowe (——) i obliczanie odwrotności dzielnika (- - -)
(2–D – uzupełnianie przybliżenia, MULT – matryca mnożąca, NORM – przesuwnik, ADD – sumator, ROUND – układ zaokrągleń, $m_1$, $m_0$ – bity części całkowitej iloczynu)

## Obliczanie odwrotności dzielnika metodą Newtona-Raphsona




kolejne przybliżenia miejsca zerowego $f(x)$ określa równanie

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

w odniesieniu do funkcji $f(x) = x^{-1} - D$ przybiera postać

$$x_{i+1} = x_i(2 - Dx_i)$$

## Obliczanie odwrotności pierwiastka metodą Newtona

liczba pierwiastkowana jest znormalizowana $\frac{1}{4} \le A < 1$.

metoda iteracyjna Newtona – równanie rekurencyjne

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

w odniesieniu do funkcji $f(x) = x^{-k} - A$ zależność przybiera postać

$$x_{i+1} = x_i - \frac{(x_i^{-k} - A)}{-kx_i^{-k-1}} = x_i\left(\tfrac{k+1}{k} - \tfrac{1}{k}x_i^k A\right)$$

jeśli $k$ =2, to iteracja określa kolejne przybliżenia odwrotności pierwiastka

$$x_{i+1} = \tfrac{1}{2}x_i(3 - x_i^2 A)$$