Organizacja i architektura komputerów [1]

Wykład 5

Piotr Patronik

9 kwietnia 2015

[1] (Prawie) dokładna kopia slajdów dr hab inż. J. Biernata

Dodawanie i odejmowanie (1)

$x_i \pm y_i \pm c_i = \pm 2c_{i+1} + s_i$

kod naturalny NB

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

kod uzupełnieniowy U2

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

$\Downarrow$

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

odejmowanie – dodanie uzupełnienia / dopełnienia z korekcją

$X - Y = X + (-Y) = X + Y + 1$

Zmiana znaku liczby w kodzie U2

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

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)

Dodawanie i odejmowanie liczb naturalnych jako rozszerzeń U2

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

$$\downarrow$$

$(s_m = c_m)$, 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

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

$$\downarrow$$

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

$$\downarrow$$

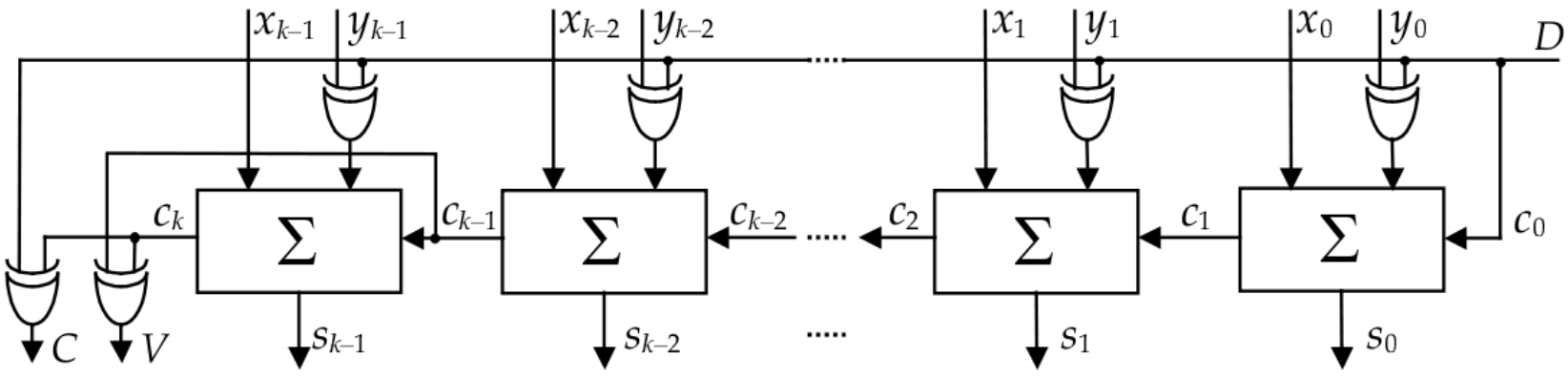
$$S = -(0+1)2^m + \sum_{i=0}^{m-1}(x_i + (1-y_i))\,2^i + 1 =$$
$$\sum_{i=0}^{m-1} s_i 2^i - (1-c_m)2^m$$

Uniwersalny sumator kaskadowy (RCA)

odejmowanie – dodanie uzupełnienia

- w systemie naturalnym – rozszerzenie wirtualne

wniosek: można skonstruować sumator uniwersalny



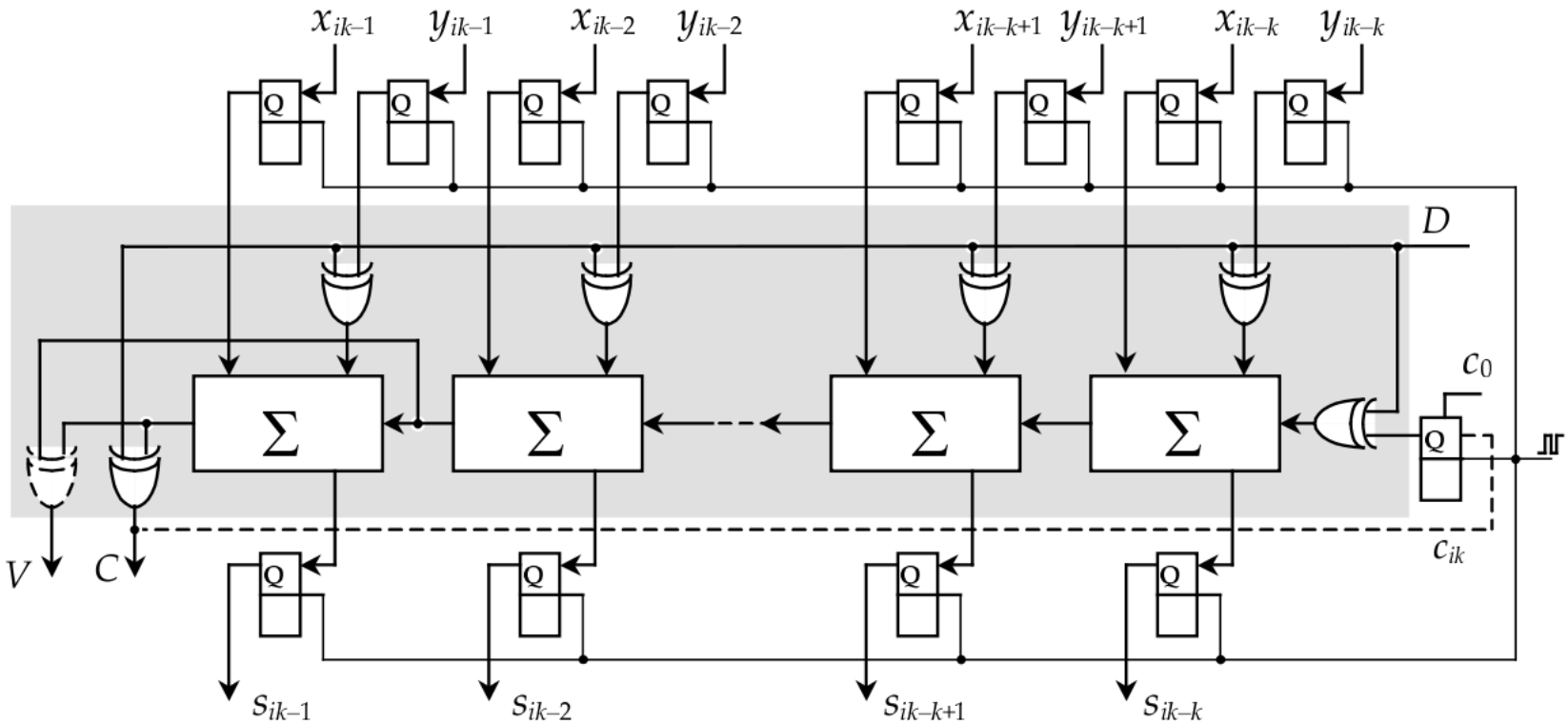
$D = 0$ – dodawanie, $D = 1$ – odejmowanie

wskaźnik poprawności (sygnalizacja przekroczenia zakresu):

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

Rozszerzenie zakresu dodawania/odejmowania

powiązanie kolejnych pozycji – bit przeniesienia

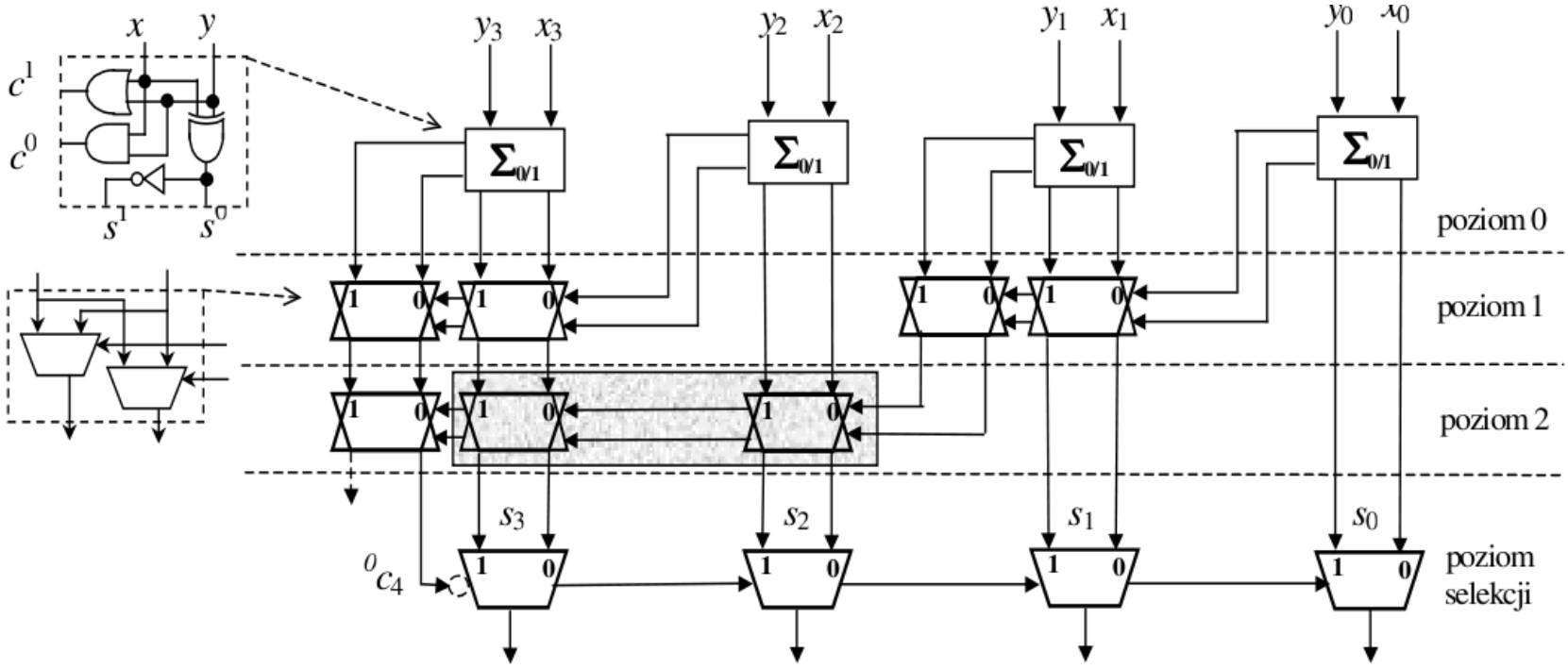


Szybkość dodawania

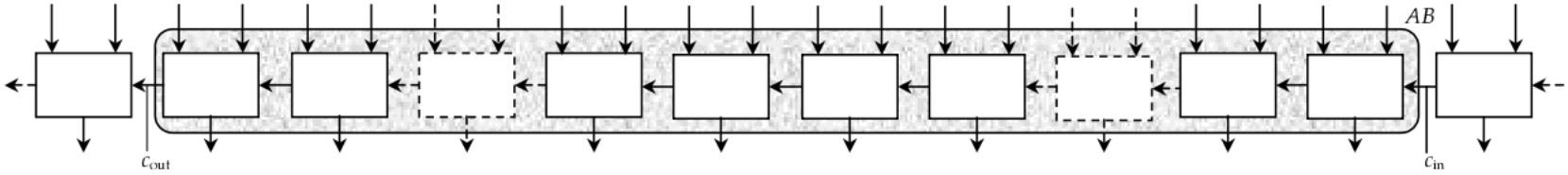
- czas dodawania zależy od *szybkości propagacji przeniesień*

rozwiązania

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

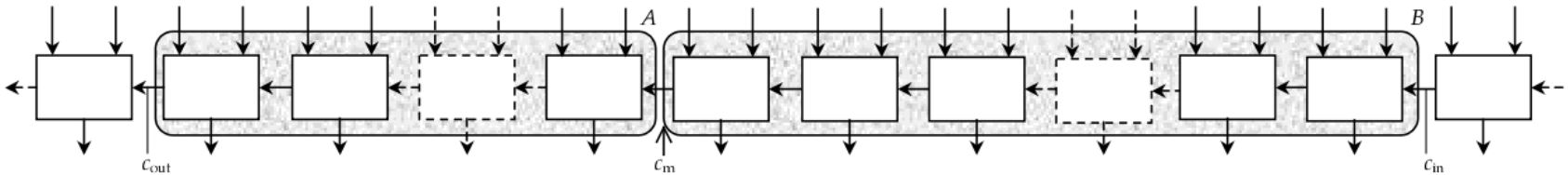


Propagacja i generowanie przeniesień – intuicje (1)



$c_{out} = 1$ jeśli

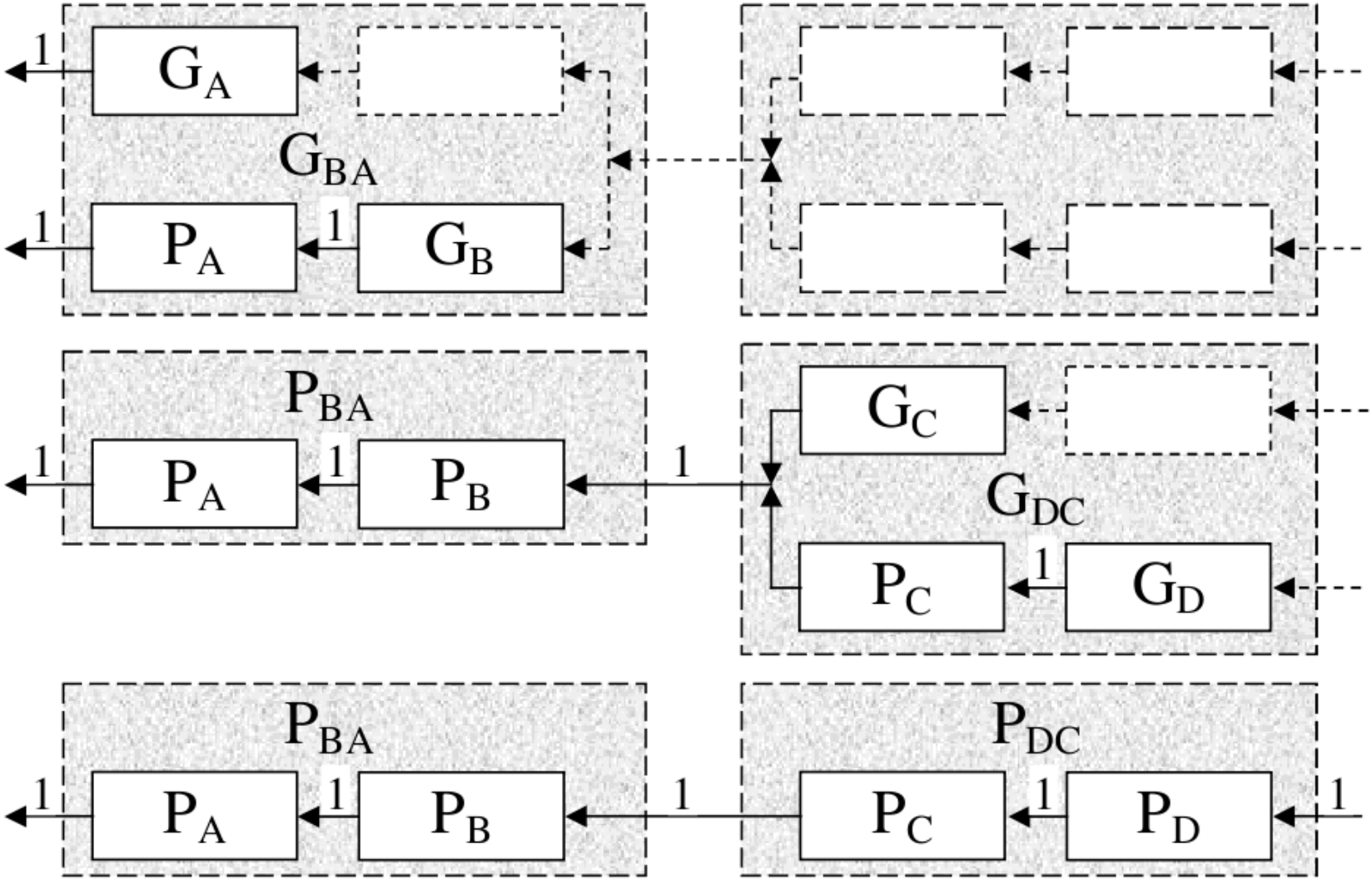
- $c_{in} = 1$ jest przesyłane przez blok AB do wyjścia c_{out}
- wewnątrz bloku AB jest tworzone $c_{out} = 1$, zaś c_{in} jest dowolne



$c_{out} = 1$ jeśli

- $c_{in} = 1$ jest przesyłane przez blok B do c_m i przez blok A do c_{out}
- wewnątrz bloku A jest wytwarzane $c_{out} = 1$, zaś c_m jest dowolne,
- wewnątrz bloku B jest wytwarzane $c_m = 1$, następnie przez blok A jest przekazywane do c_{out}

Propagacja i generowanie przeniesień – intuicje (2)



$$c_{out} = G_A + P_A G_B + P_B P_A c_{in/B} = G_{BA} + P_{BA} c_{in/B}$$
$$c_{out} = G_{BA} + P_{BA} G_{DC} + P_{BA} P_{DC} c_{in/D} = G_{DCBA} + P_{DCBA} c_{in/D}$$

Funkcje wytwarzania przeniesień i sum

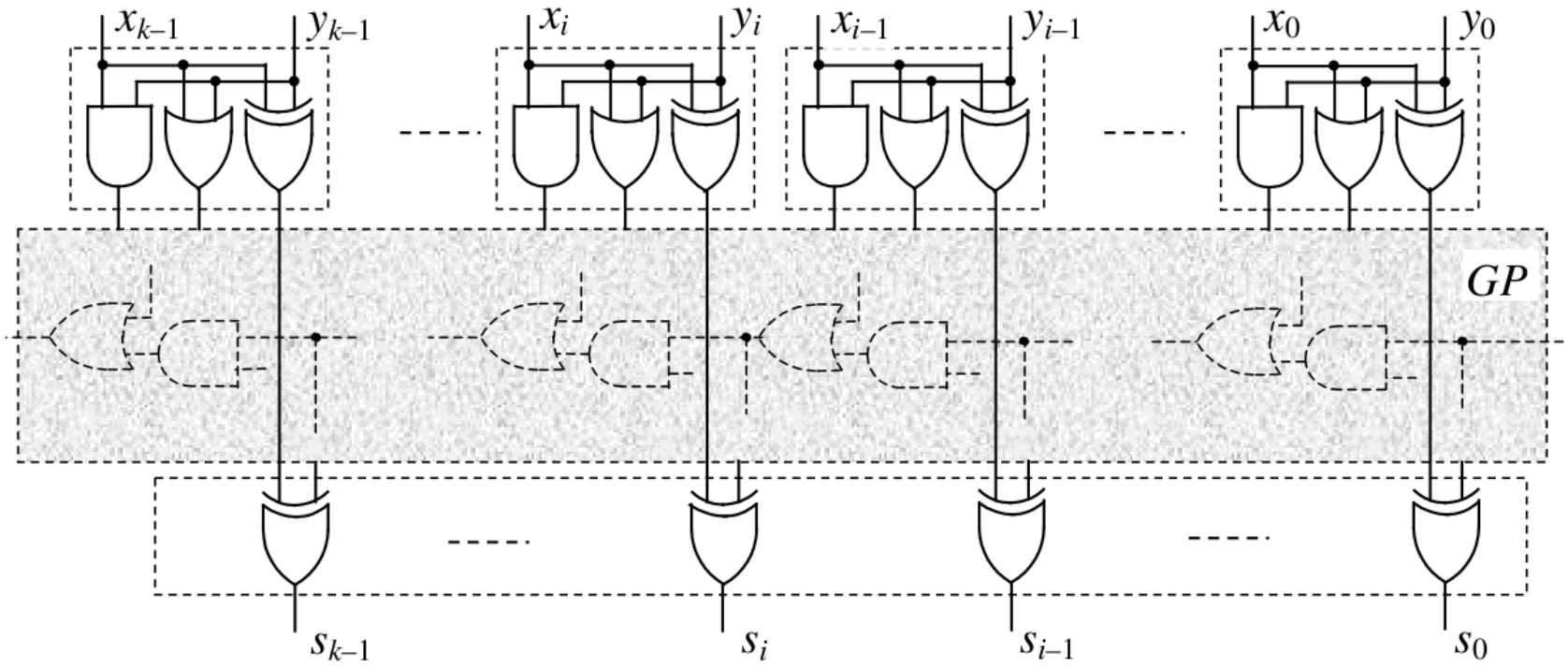
Dla bloku sumatora pomiędzy pozycjami i oraz k ($k \leqslant s \leqslant i$):
$c_{k+1} = G_{i,k} + P_{i,k} c_i$
przy tym $G_{i,k} = G_{s+1,k} + P_{s+1,k} G_{i,s}$,
$P_{i,k} = P_{i,s} P_{s+1,k}$.
Ale $G_{k,k} = g_k = x_k y_k$ i $P_{k,k} = p_k = x_k + y_k$ lub
$P_{k,k} = h_k = x_k \oplus y_k$, więc
$G_{i,k} = g_k + p_k g_{k-1} + \ldots + \prod_{j=i+1}^{k} p_j g_i$, $P_{i,k} = \prod_{j=1}^{k} p_j$
Jeśli $c_0 = 0$, to wartość sumy s_i zależy tylko od funkcji $G_{0,i-1}$ oraz h_i
$s_i = h_i \oplus c_i = h_i \oplus G_{0,i-1}$

- 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

$s_i = h_i \oplus G_{0,j-1}$



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

Szybkie dodawanie wieloargumentowe

przemienność dodawania
w systemie pozycyjnym mamy
przyśpieszenie dodawania
równoległe działania na poszczególnych pozycjach:
zamiana k argumentów o tej samej wadze (na danej pozycji)
na m argumentów o różnych wagach (zapis pozycyjny)

$$\beta^i \left(x_i^{(1)} + x_i^{(2)} + \ldots + x_i^{(k)} \right) =$$

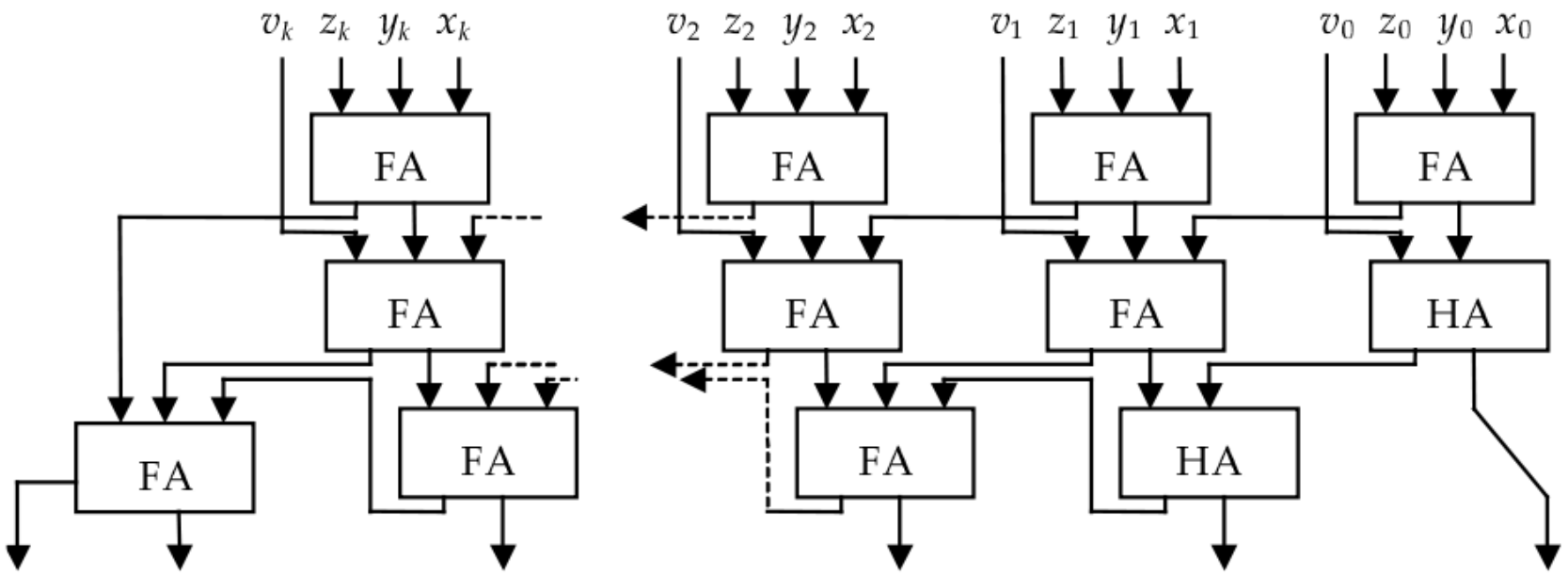
$$\beta^i \left(u_i^{(0)} + u_i^{(1)} \beta^1 + \ldots + u_i^{(m-1)} \beta^{m-1} \right)$$

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

Sumator CSA

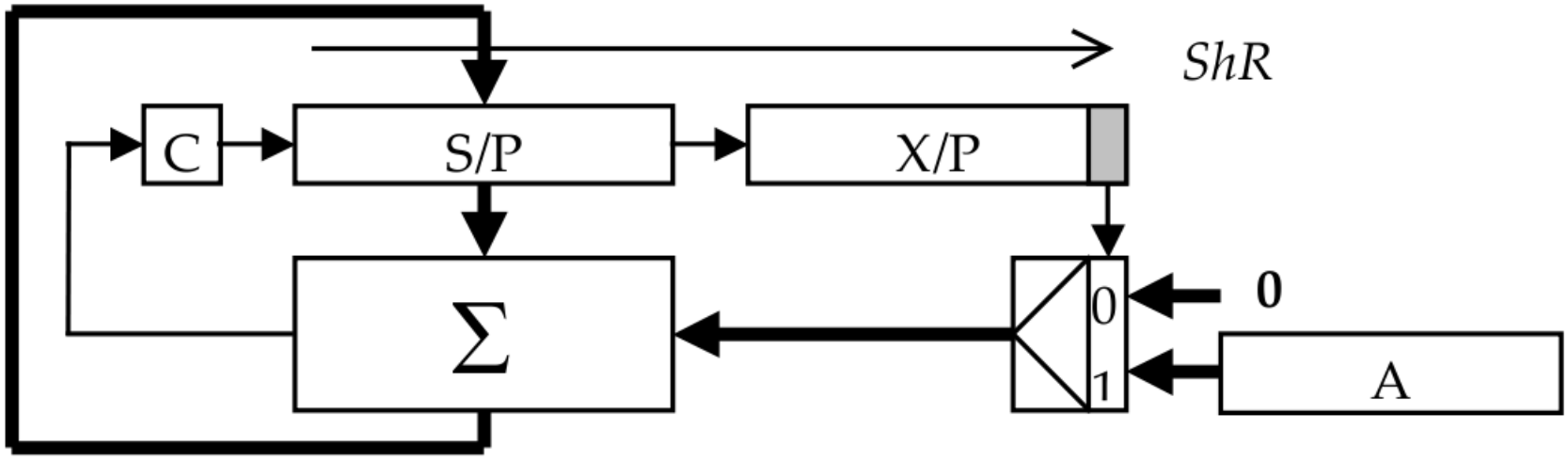
System dwójkowy:
na jednym poziomie $k = 3,\ m = 2$

Na l poziomach redukcja k_l argumentów: $2(\sqrt{2})^l \leqslant k_l \leqslant 2\left(\frac{3}{2}\right)^l$



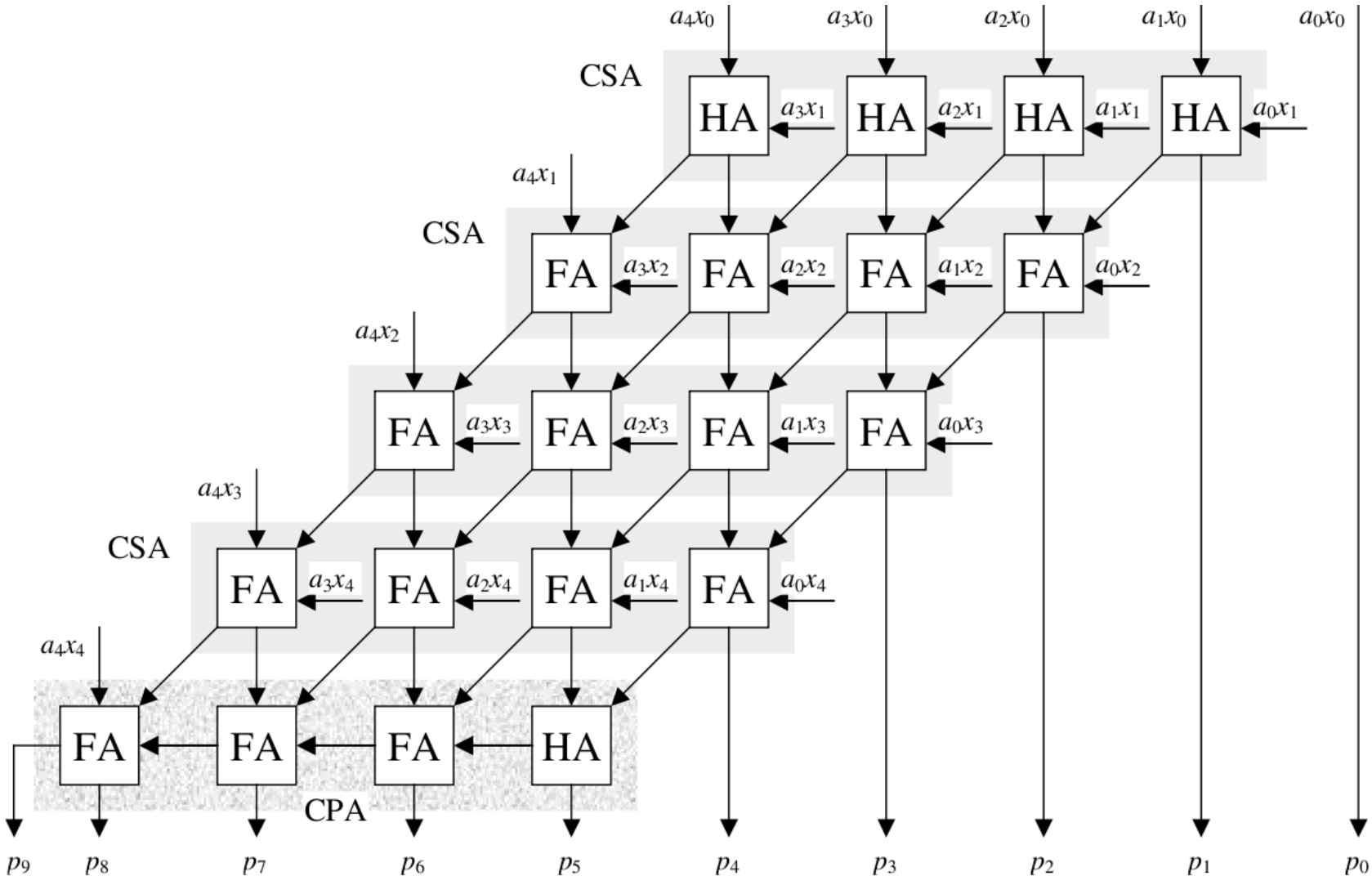
Mnożenie

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



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

Szybkie mnożenie – idea



Matryca mnożąca – liniowa struktura drzewa CSA

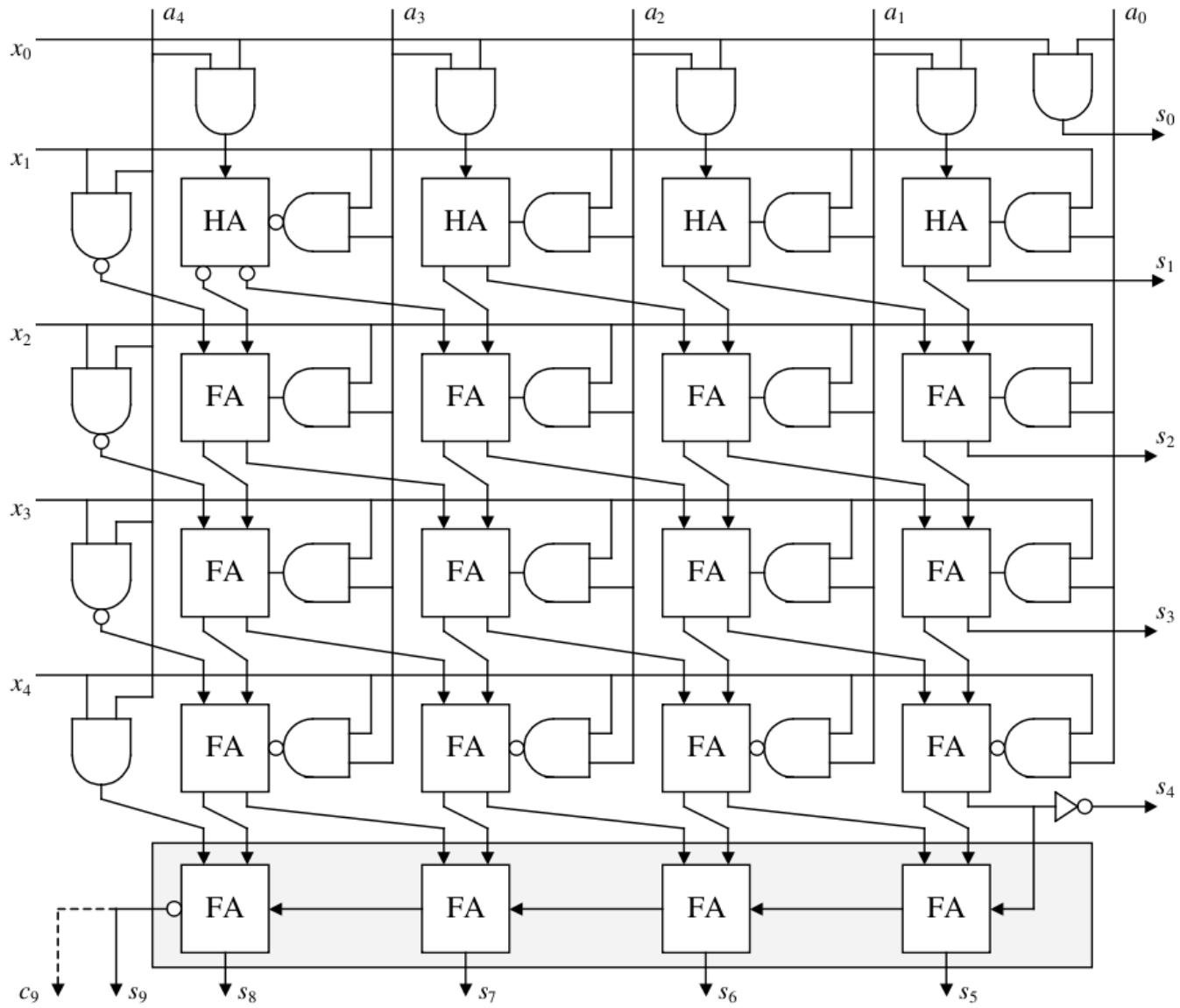
Mnożenie w dwójkowym systemie uzupełnieniowym

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

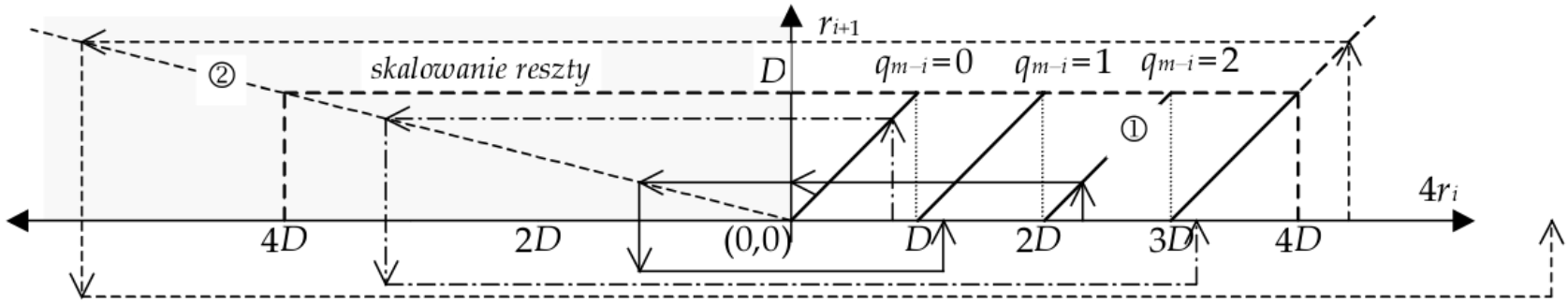
- $B_{\mathrm{U2}} = -b_{m-1}2^{m-1} + \sum_{i=0}^{m-2} b_i 2^i =$
  $-2^{m-1} + (1 - b_{m-1})2^{m-1} + \sum_{i=0}^{m-2} b_i 2^i = -2^{m-1} + B^*_{\mathrm{NB}}$
- $X_{\mathrm{U2}} A_{\mathrm{U2}} = \sum_{i=0}^{k-2} x_i 2^i A_{\mathrm{U2}} + x_{k-1}(-A_{\mathrm{U2}})2^{k-1}$
- $x_i A_{\mathrm{U2}} = -x_i a_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} x_i a_i 2^i =$
  $= -2^{m-1} + (1 - x_i a_{m-1})2^{m-1} + \sum_{i=0}^{m-2} (x_i a_i)2^i = -2^{m-1} + A x^{(i)}_{\mathrm{NB}}$

Stąd wynika, że
$X_{\mathrm{U2}} A_{\mathrm{U2}} = \sum_{i=0}^{k-1} 2^i A^{(i)}_{\mathrm{NB}} + 2^{m-1} - 2^{k+m-1}$

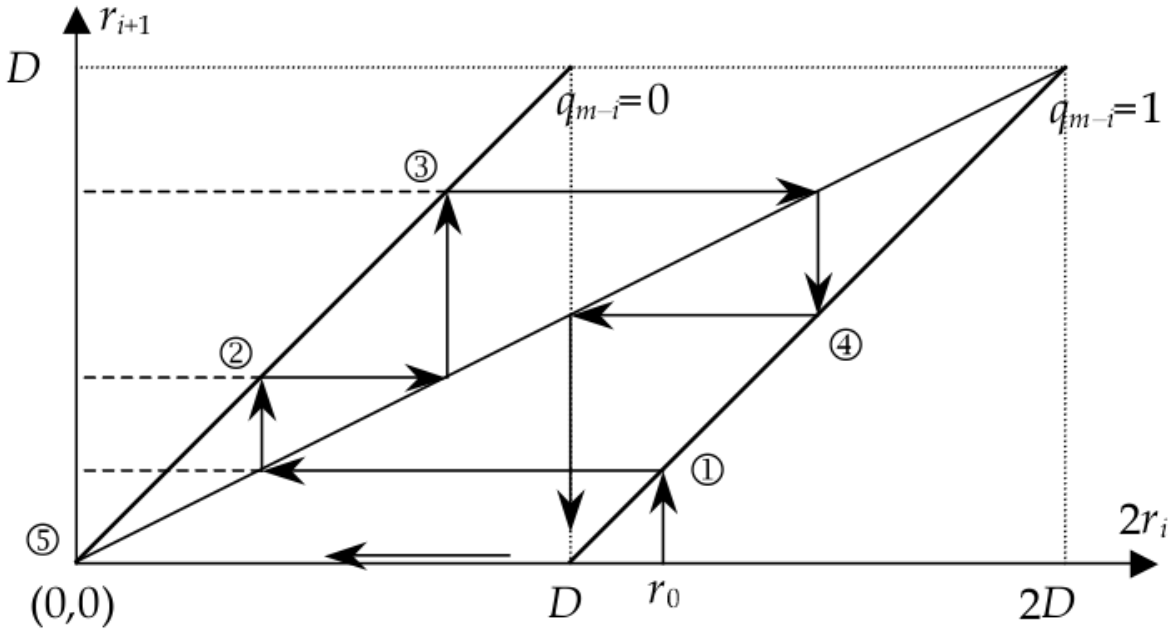
Szybkie mnożenie w kodzie uzupełnieniowym



Dzielenie



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

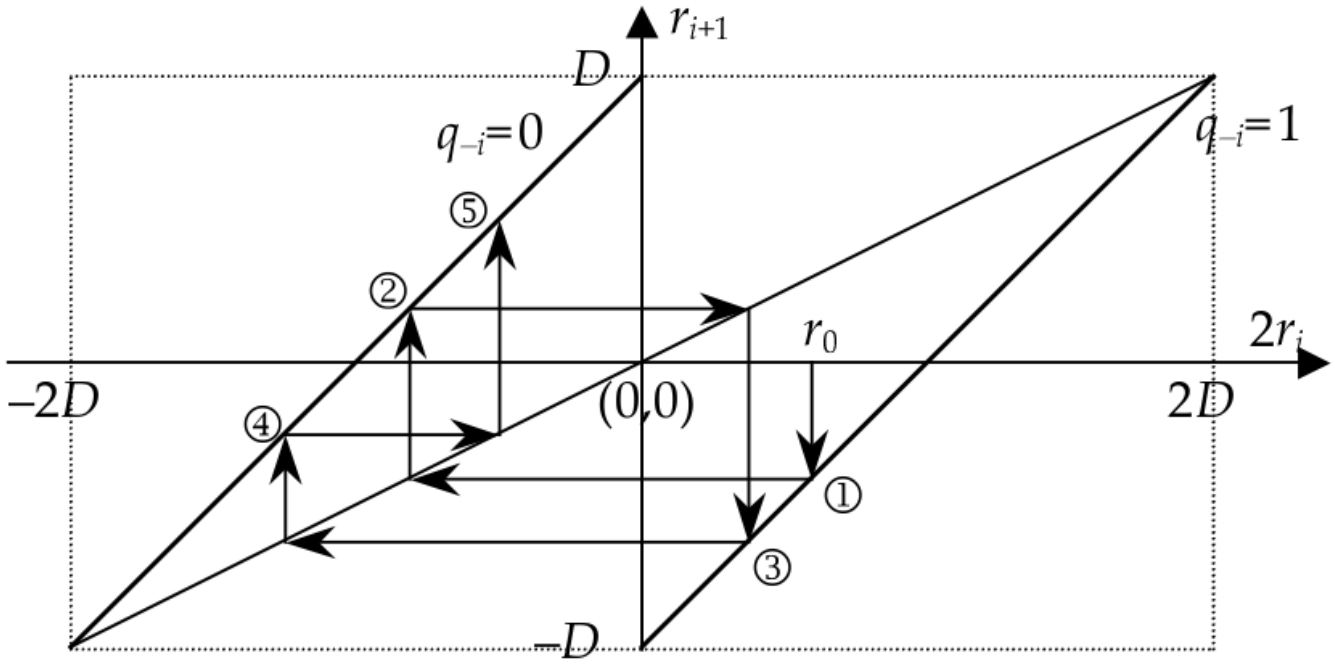


Dzielenie w systemie dwójkowym – kolejne bity ilorazu: 10010

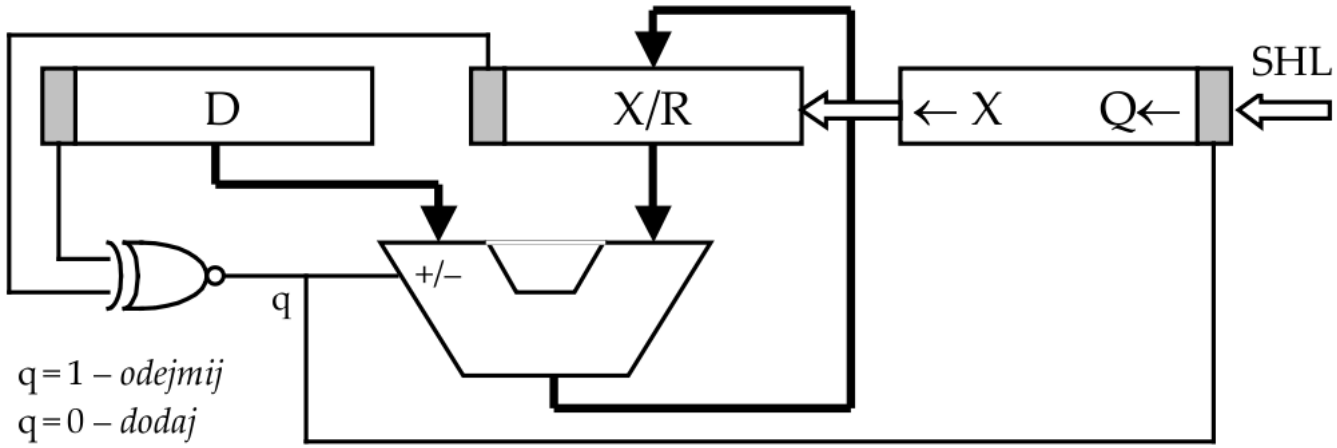
Dzielenie nieodtwarzające

- skalowanie: $|2^{-m}X| < |D| \rightarrow$ normalizacja ilorazu
  $(|D/2| < |2^{-m}X| < |D|)$
  znormalizowanym ilorazem jest $0 + \mathtt{f}(0, \mathtt{f}_{\mathtt{U2}})$ lub $-1 + \mathtt{f}(1, \mathtt{f}_{\mathtt{U2}})$
- $XD < 0 \rightarrow r_0 = 2^{-m}X + D,\ q_0 = 1;$
  $XD > 0 \rightarrow r_0 = 2^{-m}X - D,\ q_0 = 0$

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

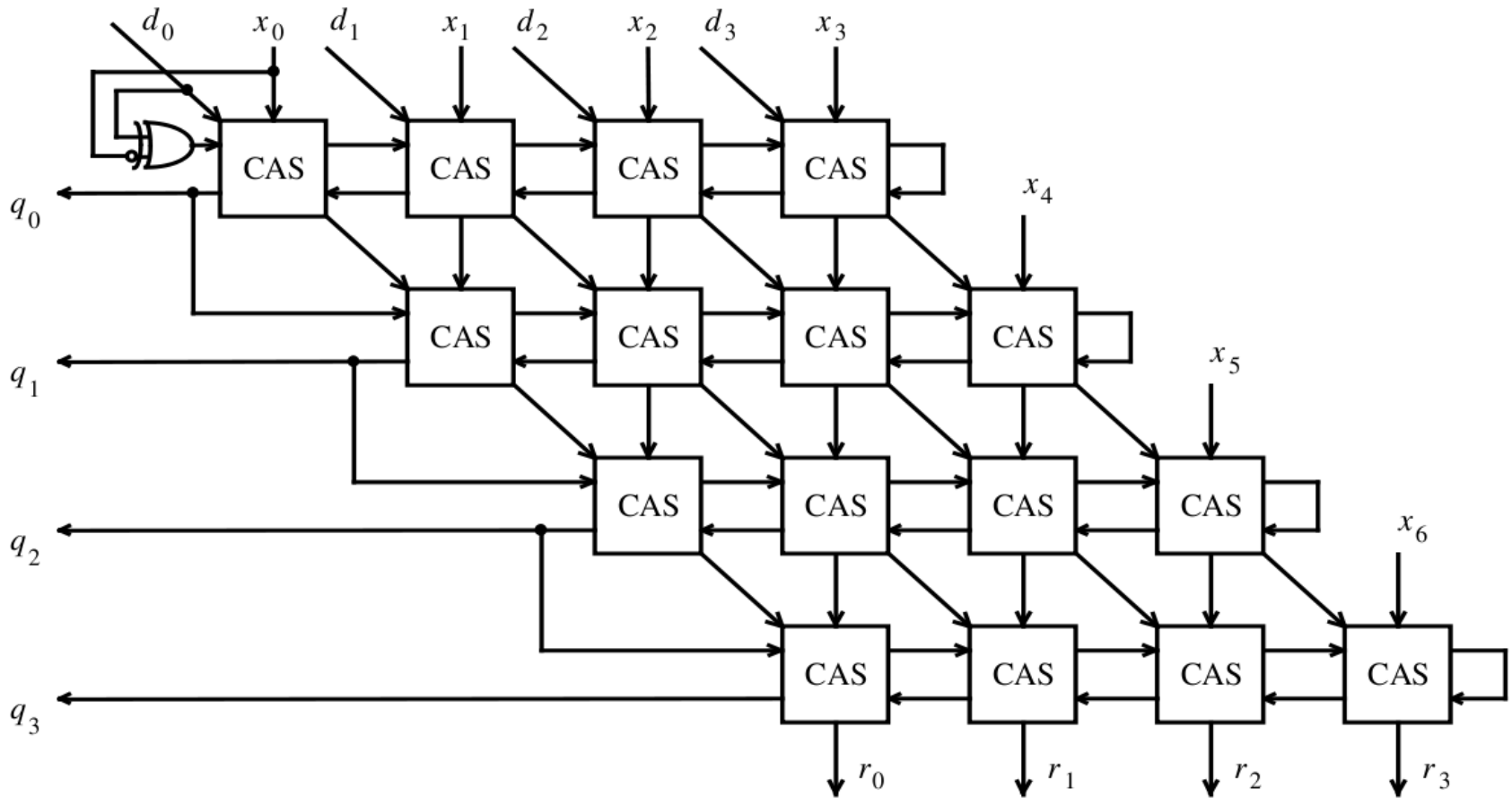


Dzielenie nieodtwarzające



- 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 754/854

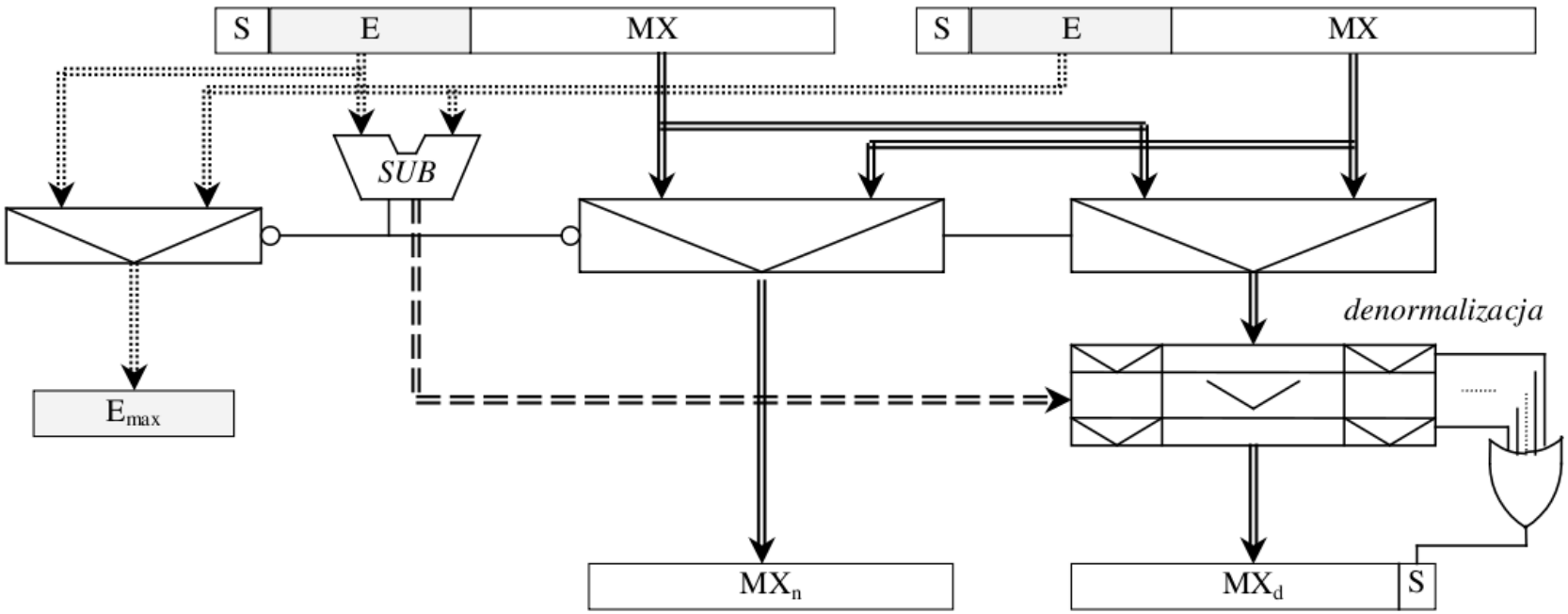
Parametr	Symbol	SINGLE	EXT-SNGL	DOUBLE	EXT-DBL
Rozmiar formatu	n	32	≥43	64	≥79
Rozmiar znacznika*	m	23 (+1)	≥32	52 (+1)	≥64
Rozmiar wykładnika	e	8	≥11	11	≥15
Obciążenie wykładnika	N	127	≥1023	1023	≥16383
Zakres wykładnika	E	[–126,+127]	[–(N–1),+N]	[–1022,+1023]	[–(N–1),+N]
Dokładność *	ulp	$2^{-23} \oplus 10^{-7}$	2^{-m+1}	$2^{-52} \oplus 10^{-15}$	2^{-m+1}
Zakres formatu	RNG	$\cong 2^{128}$ $\cong 3{,}8 \cdot 10^{38}$	$\geq 2^{1024}$	$\cong 2^{1024}$ $\cong 9 \cdot 10^{307}$	$\geq 2^{16384}$

Problemy

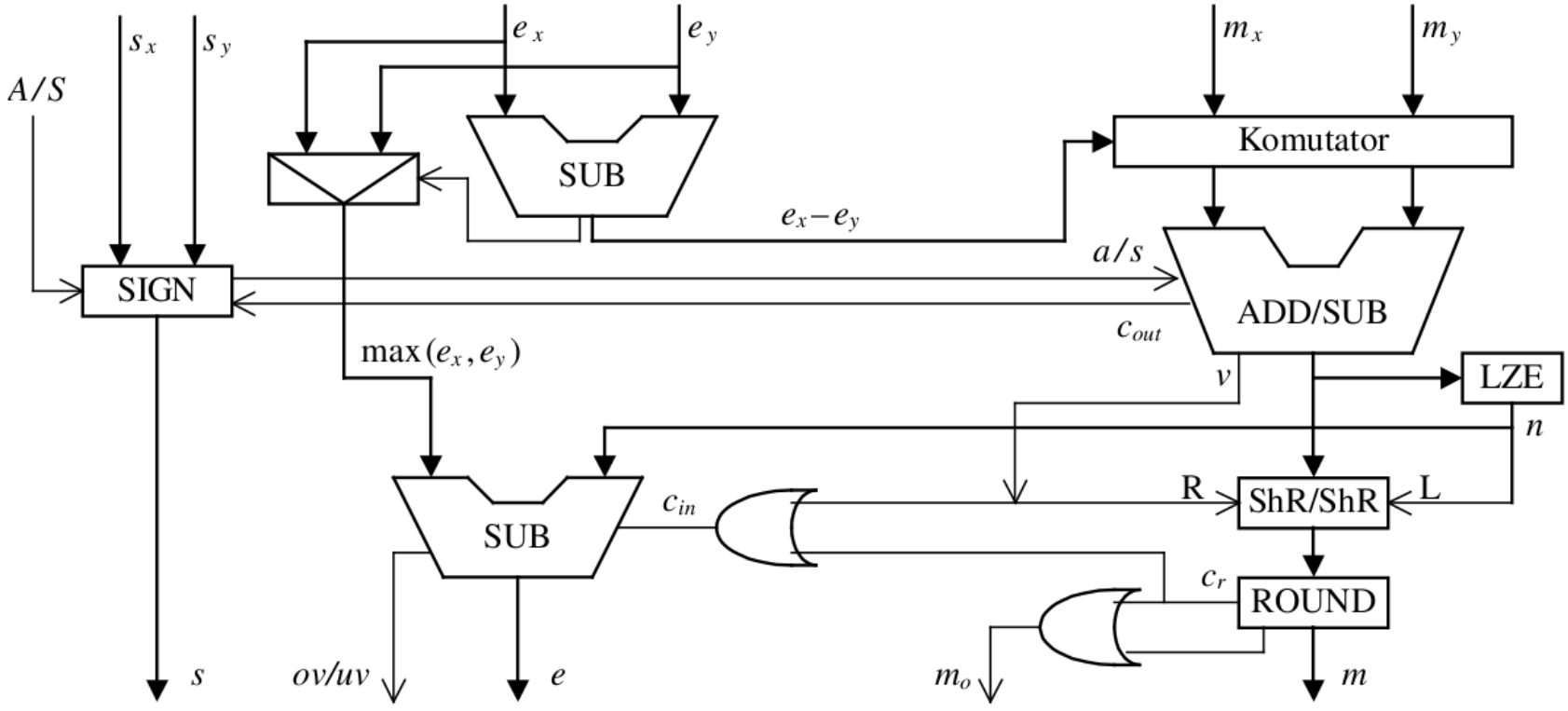
- niedokładność
- zaokrąglanie i normalizacja
- nadmiar lub niedomiar $\Rightarrow$ obsługa (skalowanie wyniku)
- nie-liczby

Wyrównanie argumentów zmiennoprzecinkowych

- Dodawanie i odejmowanie
  $F_1 \pm F_2 = M_1\beta^{E_1} \pm M_2\beta^{E_2} = \beta^{E_1}(M_1 \pm M_2\beta^{E_2-E_1})\ (E_1 > E_2)$
- zwykle konieczne wyrównanie – denormalizacja argumentu



Sumator zmiennoprzecinkowy

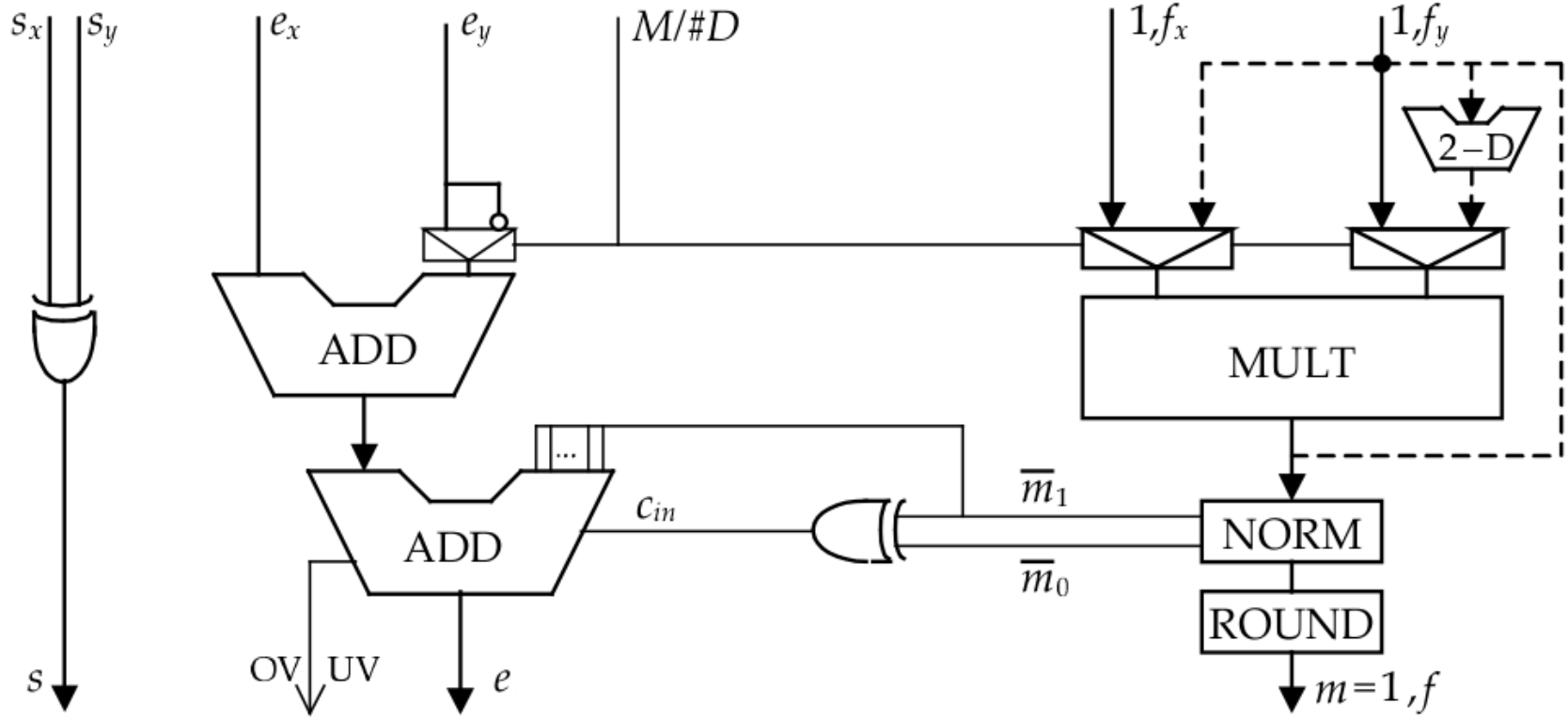


Moduł wykładnika: SIGN – generator znaku wyniku, MPX – multiplekser wyboru wykładnika wyniku, SUB – układ odejmujący wykładniki, ALIGN – sterowanie denormalizacją znaczników. *Moduł znacznika*: ADD/SUB - sumator znaczników, ShR – układ przesunięcia w prawo, LZE – koder wiodących zer, ShR/ShL – układ 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_2 – bity części całkowitej iloczynu)