SUMATORY WIELOARGUMENTOWE

Prawa łączności i przemienności dodawania

$$a+b+c+d+e+f+g+h+i+\ldots=\{[(a+b)+(g+h)]+[(e+f)+(c+d)]\}+\{[(i+\ldots$$

$$a+b+c+d+e=(((a+b)+c)+d)+e=(((c+e)+b)+d)+a=\ldots$$

Prawo łączności dodawania w systemie pozycyjnym

$$A+B+C+\ldots=\sum_{i=-m}^{k-1}a_i\beta^i+\sum_{i=-m}^{k-1}b_i\beta^i+\sum_{i=-m}^{k-1}c_i\beta^i+\ldots=\sum_{i=-m}^{k-1}(a_i+b_i+c_i+\ldots)\beta^i$$

Suma argumentów jednocyfrowych jest wielocyfrowa (w zapisie pozycyjnym):

$$a_i+b_i+c_i+\ldots=u_i^{(m)}\beta^m+\ldots+u_i^{(2)}\beta^2+u_i^{(1)}\beta+u_i^{(0)}$$

- suma **wielocyfrowa** $\{u_i^{(m-1)},\ldots,u_i^{(2)},u_i^{(1)},u_i^{(0)}\}$ w kolumnie cyfr o ustalonej wadze ma wagę taką jak waga pozycji (kolumny), a jej rozmiar jest logarytmicznie zależny od liczby argumentów n ($m=\lfloor \log_\beta n \rfloor$)

Łączność i przemienność dodawania w systemie pozycyjnym

$$X^{(1)}+X^{(2)}+\ldots+X^{(k)}=\sum_{p=1}^{k}\sum_{i=-q}^{n-1}x_i^{(p)}\beta^i=\sum_{i=-q}^{n-1}\beta^i\sum_{p=1}^{k}x_i^{(p)}=\sum_{i=-q}^{n-1}\beta^i\left(\sum_{r=0}^{m}u_{i-r}^{(r)}\beta^r\right)$$

Dodawanie wieloargumentowe jednopozycyjne w systemach naturalnych

$$x_i^{(1)} + x_i^{(2)} + x_i^{(3)} + \ldots + x_i^{(n)} = u_i^{(m)}\beta^m + \ldots + u_i^{(2)}\beta^2 + u_i^{(1)}\beta + u_i^{(0)}, \text{ gdzie } x_i^{(j)}, u_i^{(j)} \in \{0,1,\ldots,\beta-1\}$$

Aby suma była najwyżej m-cyfrowa liczba składników k musi spełniać warunek:

$$\sum_{j=1}^{k} x_i^{(j)} \le \sum_{j=1}^{k} (\beta-1) = k(\beta-1) \le \beta^m - 1, \text{gdzie } 0 \le x_i^{(j)} \le \beta-1$$

$$\text{czyli } k \le (\beta^m - 1)/(\beta-1) = \beta^{m-1} + \beta^{m-2} + \ldots + \beta + 1 = 11\ldots11_\beta$$

dodawanie można wykonać dwuetapowo:

- obliczyć wielopozycyjne sumy przejściowe (w dowolnej kolejności)
- dodać liczby wielocyfrowe skomponowane z sum przejściowych

$\rightarrow$ jeśli liczba składników jest $\le\beta+1$, to suma jest dwucyfrowa i wynosi

$$\{v_{i+1}, u_i\} = \{r,\ x_i^{(1)} + x_i^{(2)} + \ldots + x_i^{(\beta+1)} - r\beta\} \text{ gdy } 0 \le x_i^{(1)} + x_i^{(2)} + \ldots + x_i^{(\beta+1)} - r\beta < \beta$$

Dodawanie wieloargumentowe w systemach naturalnych

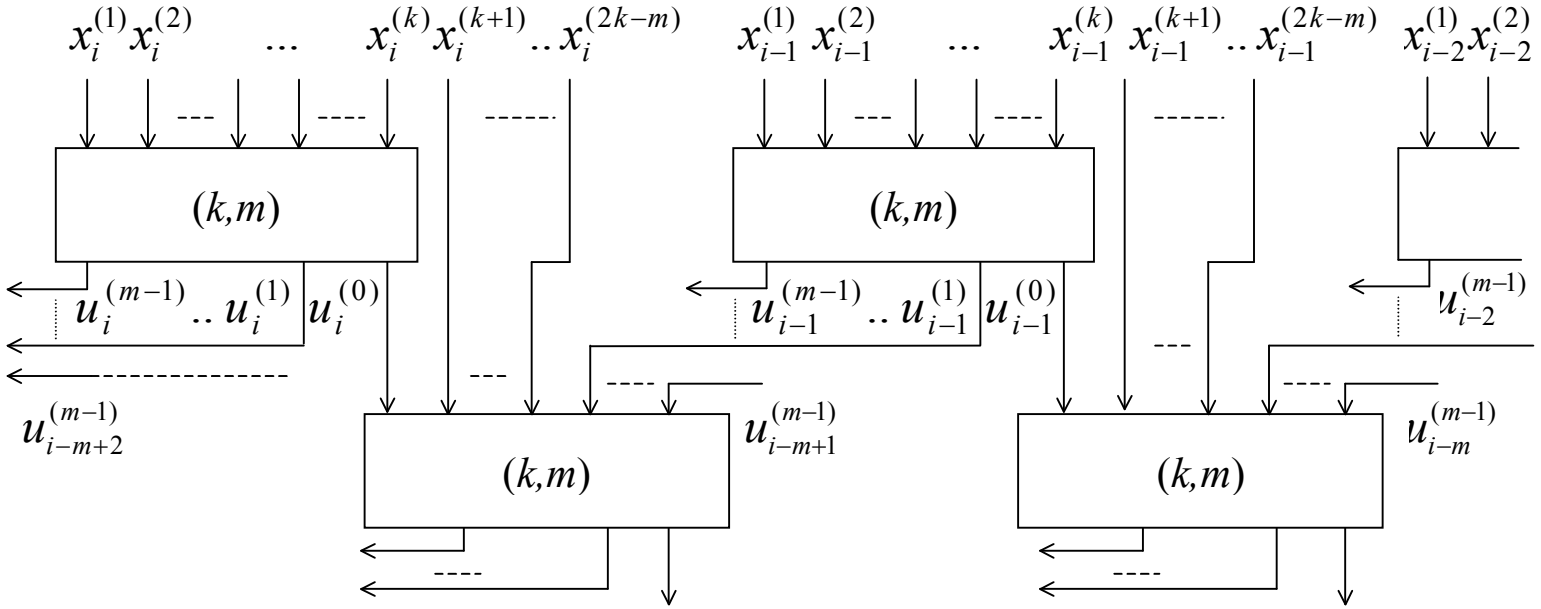
→ jeśli liczba argumentów $k>\beta+1$, to dodawanie można wykonać etapami

$>\beta+1$ argumentów		(0)	(0)	a_{k-1}	a_{k-2}	…	a_{-m+3}	a_{-m+2}	a_{-m+1}	a_{-m}
		(0)	(0)	b_{k-1}	b_{k-2}	…	b_{-m+3}	b_{-m+2}	b_{-m+1}	b_{-m}
		(0)	(0)	c_{k-1}	c_{k-2}	…	c_{-m+3}	c_{-m+2}	c_{-m+1}	c_{-m}
		(0)	(0)	d_{k-1}	d_{k-2}	…	d_{-m+3}	d_{-m+2}	d_{-m+1}	d_{-m}
		…	…	…	…	…	…	…	…	…
	+	(0)	(0)	p_{k-1}	p_{k-2}	…	p_{-m+3}	p_{-m+2}	p_{-m+1}	p_{-m}
		…	…	…	…	…	…	…	…	…
		…	…	…	…	…	…	…	…	…
$\leq\beta+1$ arg.		(0)	(0)	$^{(0)}x_{k-1}$	$^{(0)}x_{k-2}$	…	$^{(0)}x_{-m+3}$	$^{(0)}x_{-m+2}$	$^{(0)}x_{-m+1}$	$^{(0)}x_{-m}$
		(0)	$^{(1)}x_{k-1}$	$^{(1)}x_{k-2}$	…	$^{(1)}x_{-m+3}$	$^{(1)}x_{-m+2}$	$^{(1)}x_{-m+1}$	$^{(1)}x_{-m}$	(0)
		$^{(2)}x_{k-1}$	$^{(2)}x_{k-2}$	…	$^{(2)}x_{-m+3}$	$^{(2)}x_{-m+2}$	$^{(2)}x_{-m+1}$	$^{(2)}x_{-m}$	(0)	(0)
	+	…	…	…	…	…	…	…	…	…
2 arg	…	$^{(0)}u_{k+1}$	$^{(0)}u_{k}$	$^{(0)}u_{k-1}$	$^{(0)}u_{k-2}$	…	$^{(0)}u_{-m+3}$	$^{(0)}u_{-m+2}$	$^{(0)}u_{-m+1}$	$^{(0)}x_{-m}$
	…	$^{(1)}u_{k}$	$^{(1)}u_{k-1}$	$^{(1)}u_{k-2}$	…	$^{(1)}u_{-m+3}$	$^{(1)}u_{-m+2}$	$^{(1)}u_{-m+1}$		
	…	S_k	S_k	S_{k-1}	S_{k-2}		S_{-m+3}	S_{-m+2}	$^{(0)}u_{-m+1}$	$^{(0)}x_{-m}$

Redukcja argumentów w drzewie CSA

sumator (k,m) – układ obliczający m-pozycyjną sumę k liczb jednocyfrowych

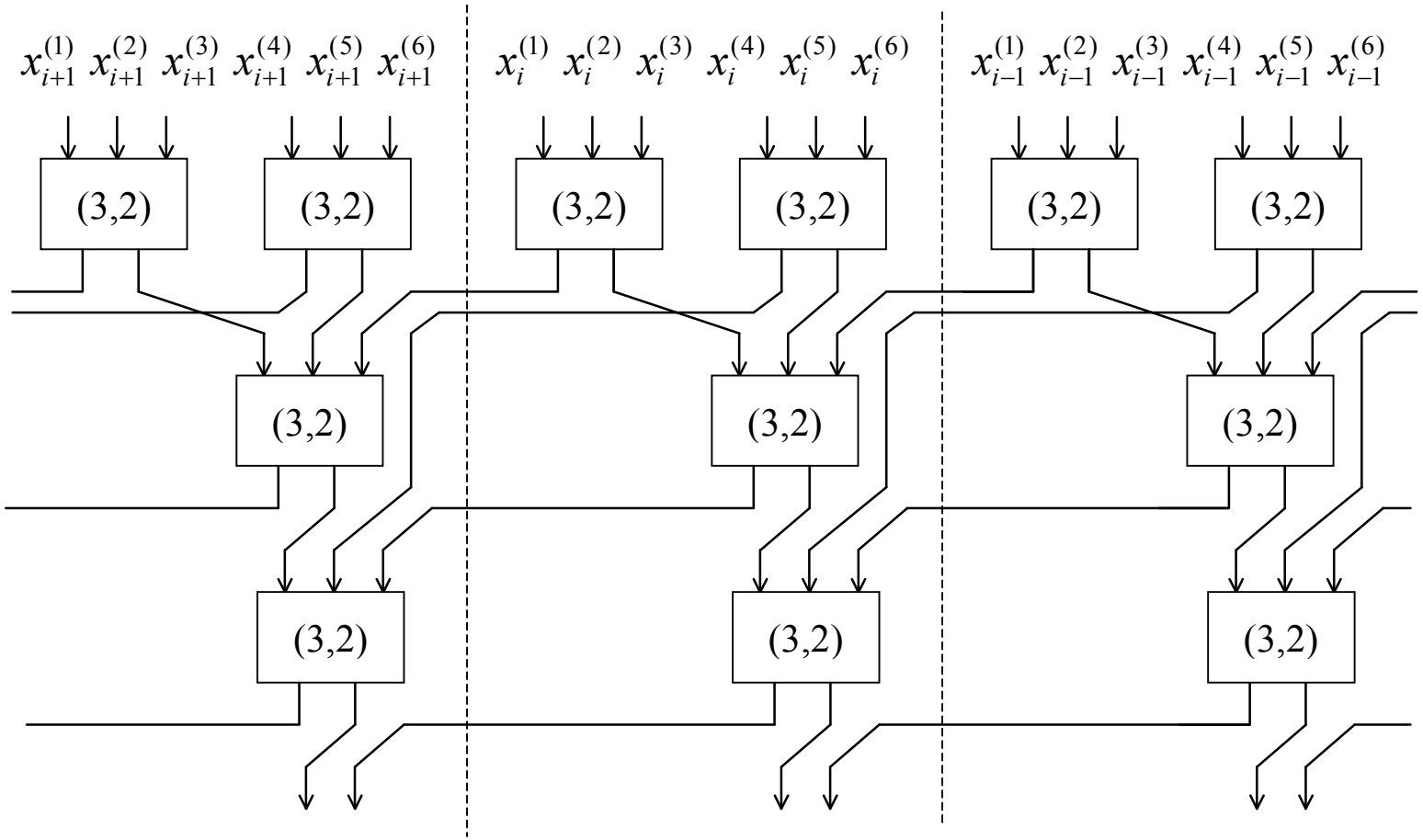
$$m = \left\lfloor \log_{\beta}[k(\beta-1)+1] \right\rfloor$$



Struktura redukcji argumentów w drzewie CSA zbudowanym z modułów (k,m)

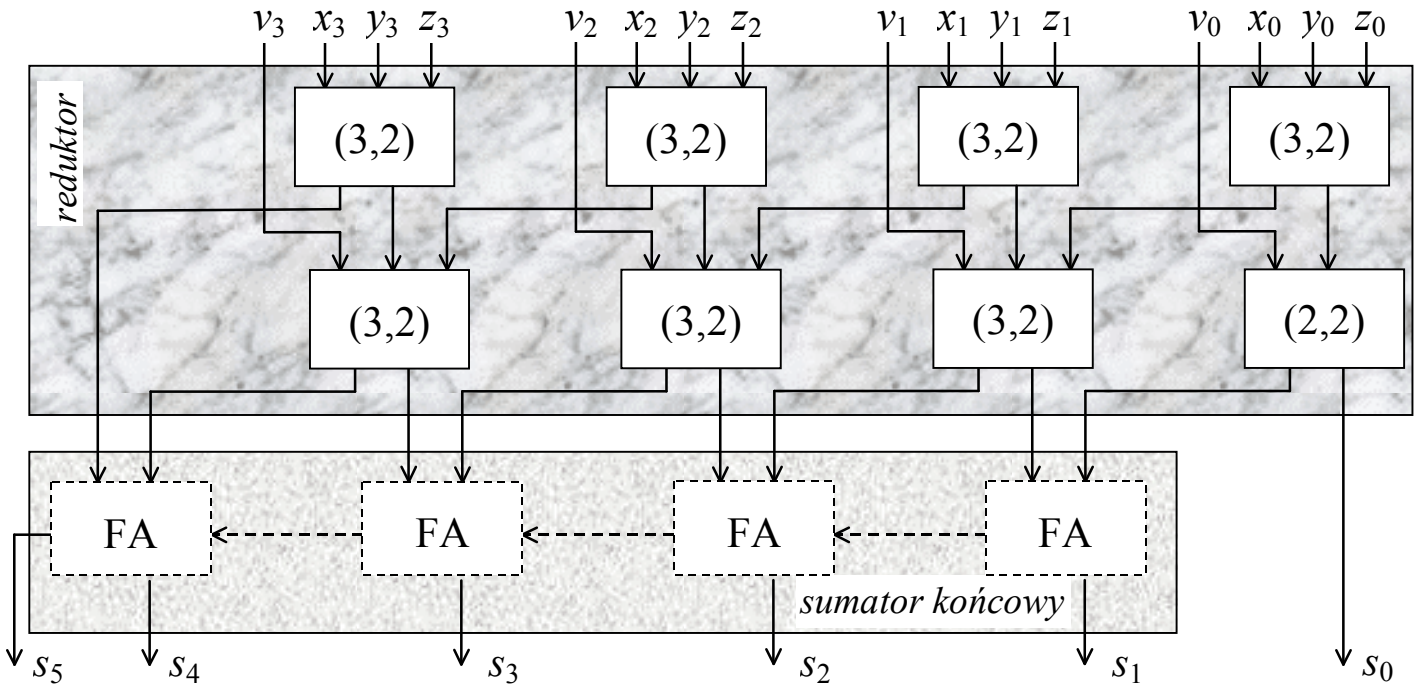
$$X^{(1)} + X^{(2)} + \ldots + X^{(k)} = \sum_{p=1}^{k} \sum_{i=-q}^{n-1} x_i^{(p)} \beta^i = \sum_{i=-q}^{n-1} \beta^i \sum_{p=1}^{k} x_i^{(p)} = \sum_{i=-q}^{n-1} \beta^i \left(\sum_{r=0}^{m} u_{i-r}^{(r)} \beta^r \right)$$

Redukcja argumentów w dwójkowym drzewie CSA



Skala redukcji operandów w wielopoziomowym dwójkowym drzewie CSA

Dwójkowe sumatory wieloargumentowe (CSA)



Czteropozycyjny sumator czterooperandowy CSA

czas dodawania = czas redukcji + czas dodawania końcowego

Redukcja *k* dwójkowych operandów *n*-pozycyjnych (CSA)

Pojedynczy układ (3,2) redukuje dokładnie jeden operand 1-bitowy

→ do redukcji k operandów n-bitowych potrzeba $n(k–2)$ układów (3,2)

→ k_L operandów na poziomie $L \Rightarrow k_{L+1} \le k_L + \lfloor k_L / 2 \rfloor$ na poziomie L+1

- jeden poziom CSA redukuje 3 operandy do 2 – skala redukcji 3/2
- dwa poziomy CSA redukują 4 operandy do 2 – skala redukcji $\sqrt{2}$
- trzy poziomy CSA redukują 6 operandów do 2 – skala redukcji $\sqrt[3]{3}$

$$2(\sqrt{2})^L \le k_L \le 2(\tfrac{3}{2})^L$$

(*lepsza ocena*) $\quad 2(\sqrt[3]{3})^L \le k_L \le 2(\tfrac{3}{2})^L \qquad (L \ge 3)$

(*niezłe oszacowanie*) $\quad k_L \cong (\tfrac{3}{2})^L + (\sqrt[3]{3})^L$

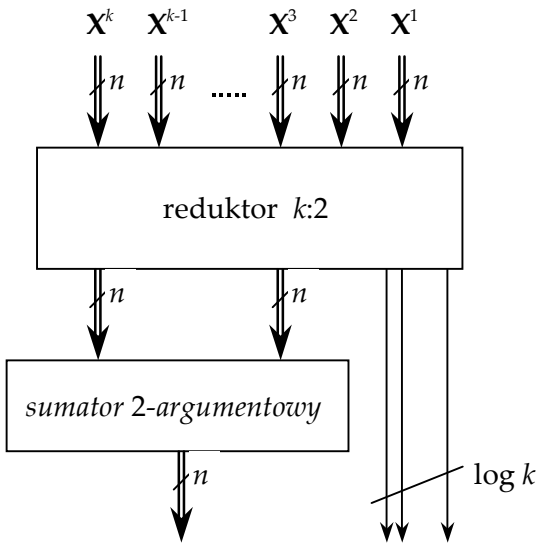
Redukcja liczby operandów w wielopoziomowej strukturze CSA

liczba poziomów L	1	2	3	4	5	6	7	8	9	10
$2(\frac{3}{2})^L$	3	4	6	10	15	22	34	51	76	115
maksymalna liczba operandów	**3**	**4**	**6**	**9**	**13**	**19**	**28**	**42**	**63**	**94**
$2(\sqrt[3]{3})^L \cong 2 \cdot 1{,}44224957^{\,L}$	3	~~5~~	6	9	13	18	26	38	54	78
$2(\sqrt{2})^L \cong 2 \cdot 1{,}41421356^{\,L}$	3	4	6	8	12	16	23	32	46	65

Konstrukcja wieloargumentowego sumatora CSA (1)

- poprawna reprezentacja sumy k argumentów n-bitowych
  $$0 \le X_1, X_2, ..., X_k \le 2^n - 1 \Rightarrow 0 \le X_1 + X_2 + ... + X_k \le k(2^n - 1) < 2^{n+\log k} - 1$$
  wymaga wytworzenia $\log k$ dodatkowych pozycji
- reduktor CSA zawiera $nk - 2n = n(k-2)$ elementów (3,2)
- reduktor ma głębokość $L = O(\log k)$:

$$1{,}7095 \log \tfrac{k}{2} = (\log 1{,}5)^{-1} \log \tfrac{k}{2} \le L \le 3(\log 3)^{-1} (\log \tfrac{k}{2}) = 1{,}8928 (\log \tfrac{k}{2})$$



sumator wieloargumentowy CSA

Konstrukcja wieloargumentowego sumatora CSA (2)

Konstrukcja jest rekurencyjna typu *top-down*

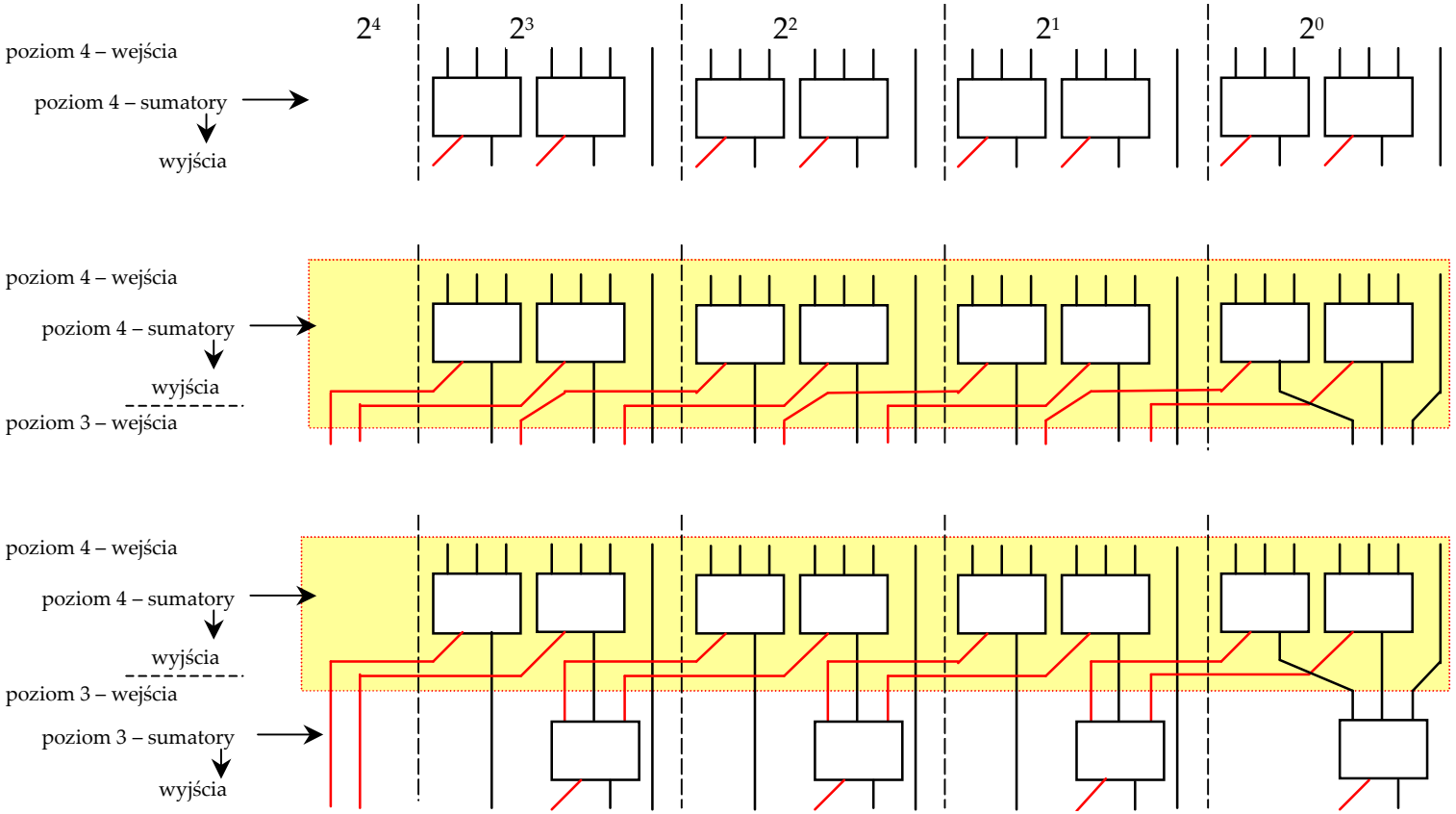
1. każde 3 sygnały wejściowe **o tej samej wadze** przyłącz do wejść modułu (3,2)
2. sygnał nieprzyłączony przekaż na niższy poziom CSA lub opcjonalnie
   parę sygnałów **o tej samej wadze** przekształć przez półsumator HA (układ (2,2))
3. wytwórz wyjścia wszystkich modułów (3,2) (lub (2,2))
   – pamiętaj, że wyjścia *s* (sumy) i *c* (przeniesienia) mają **różne wagi**!
4. w poszczególnych kolumnach zbierz sygnały **o jednakowych wagach**
5. powtarzaj 1–4 dopóki w jakiejś kolumnie pozostało więcej niż 2 sygnały
6. dołącz sumator końcowy (najlepiej szybki: CLA, PPA, COSA)

Konstrukcja typu *bottom-up* (*drzewo Wallace'a*) jest trudniejsza,
bo w skrajnych kolumnach jest mniej argumentów do redukcji

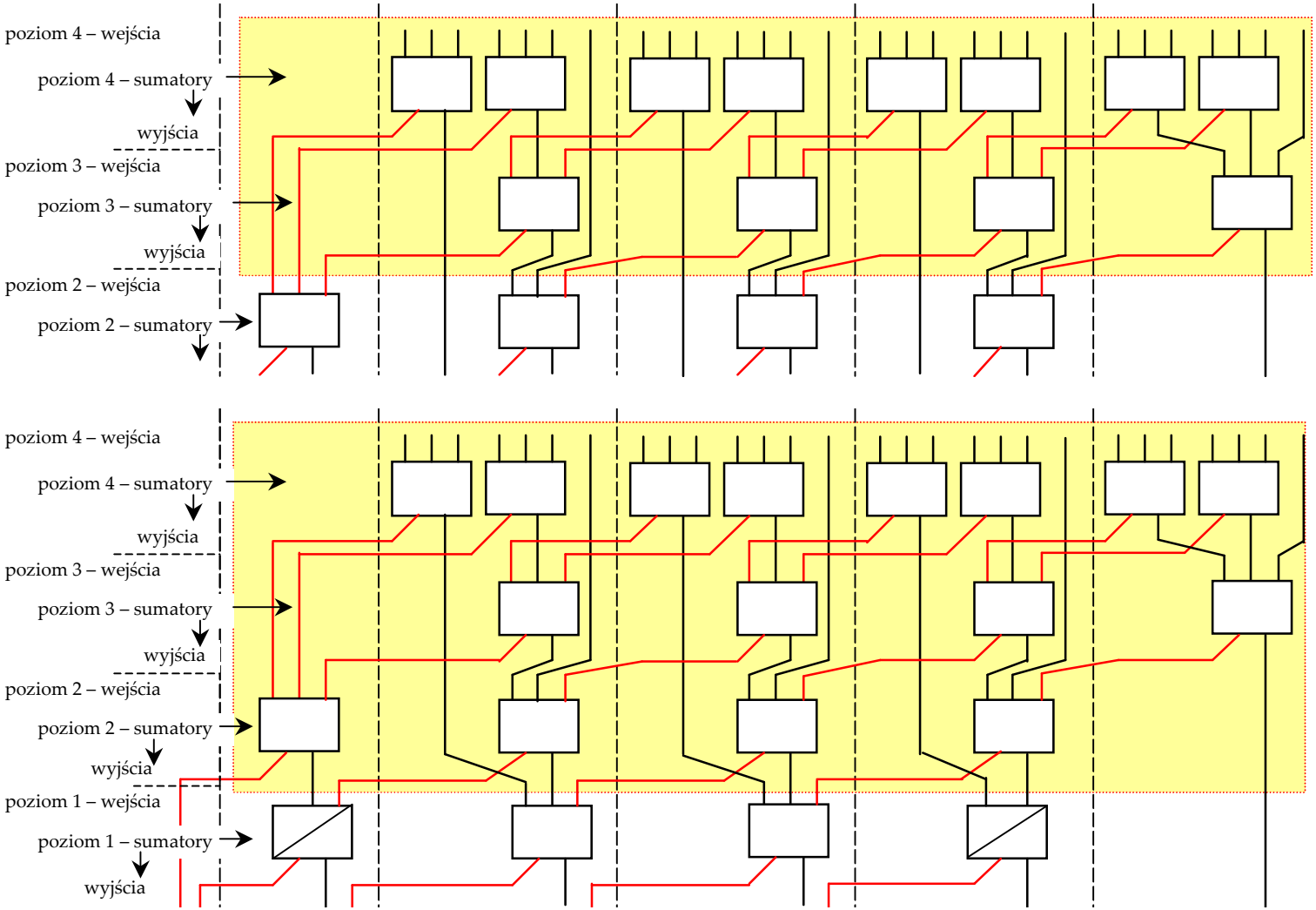
UWAGA:
Zwrotne przekazywanie sygnału na poprzedni poziom redukcji
jest **poważnym błędem** (sprzeczne z ideą szybkiej redukcji argumentów)

Schemat konstrukcji wielopoziomowego reduktora CSA (1)

przykład – 7 argumentów 4-bitowych (w=3+3, L=4)



Schemat konstrukcji wielopoziomowego reduktora CSA (2)



Dwójkowe sumatory wieloargumentowe kodu U2 (1)

Wymagany zakres końcowej sumy jest o $m=\lceil \log_2 k \rceil$ bitów większy, bo

$$-2^{n-1} \le X_1, X_2, ..., X_k \le 2^{n-1}-1 \Rightarrow -2^{n+\log k-1} \le X_1 + X_2 + ... + X_k < 2^{n+\log k-1}$$

Podstawowy algorytm dodawania w kodzie U2 wymaga użycia $m=\lceil \log_2 k \rceil$ bitów rozszerzenia lewostronnego dla każdego z k argumentów, a w konsekwencji:

- drzewo CSA zawierałoby $k\cdot(n+m)$ argumentów bitowych
- $k\cdot\lceil \log_2 k \rceil$ wejść stanowiłyby powielone bity rozszerzenia lewostronnego
- konieczne rozgałęzienie bitu wiodącego każdego spośród k argumentów

Alternatywa: przekodowanie argumentów

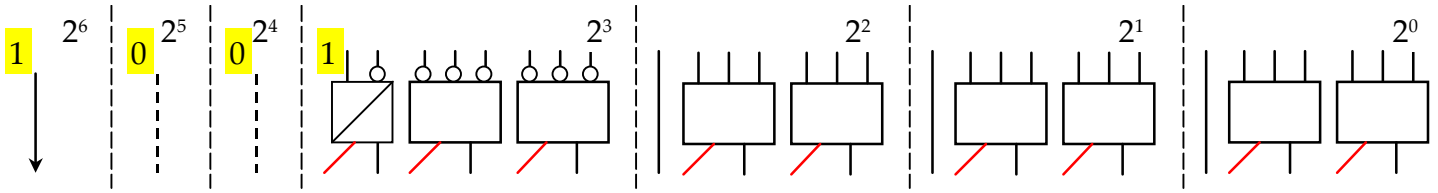
każdy argument n-bitowy w kodzie U2 można zapisać jako sumę liczby dodatniej (w kodzie NB) i stałej ujemnej

$$-x_{n-1}2^{n-1} + \sum_{i=0}^{k-2} x_i 2^i = -2^{n-1} + \overline{x}_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

gdzie $\overline{x}_{n-1} = 1 - x_{n-1}$ – dopełnienie (negacja) bitu wiodącego

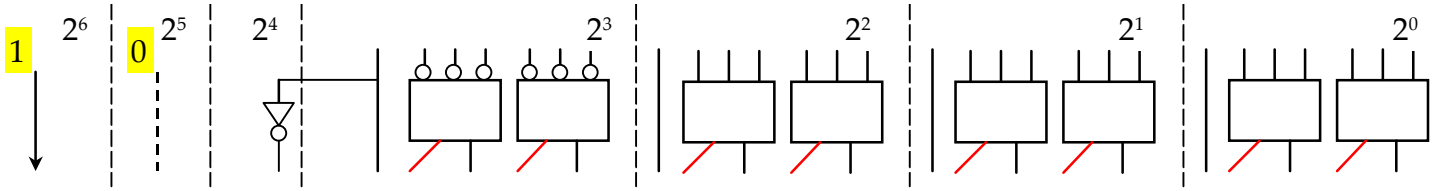
Dwójkowe sumatory wieloargumentowe kodu U2 (2)

Przekodowanie każdego argumentu $\{x_{n-1}, x_{n-2}, ..., x_1, x_0\}$ danego w kodzie U2 na sumę liczby naturalnej o reprezentacji $\{\overline{x}_{n-1}, x_{n-2}, ..., x_1, x_0\}$ i ujemnej stałej -2^n pozwala zastąpić użycie rozszerzeń lewostronnych przez dopełnianie (negację) wiodących bitów argumentów i dodanie stałej korekcyjnej $-k \cdot 2^{n-1}$:



Modyfikacja drzewa CSA do dodawania k=7 liczb n=4-bitowych w kodzie U2

UWAGA: Ponieważ $...01...11 + ...00x = ... \, x \, \overline{x} \, ... \overline{x} \, \overline{x}$, więc schemat można uprościć:



Uproszczone drzewo CSA do dodawania k=7 liczb n=4-bitowych w kodzie U2

Zliczanie jedynek (lub zer) w słowie *n*-bitowym

(prawie) trzykrotna redukcja na I poziomie sumatora

- jeśli $n=3k$, to na II poziomie jest k operandów 2-bitowych
- jeśli $n \neq 3k$, to na II poziomie jest $\lceil n/3 \rceil$ operandów 2-bitowych

Parametry układu (bez 2-bitowego sumatora wyjściowego)

- liczba modułów CSA – $n-2$
- liczba poziomów CSA – 1+ liczba poziomów redukcji $\lceil n/3 \rceil$ operandów, czyli
$$\frac{\log\lceil n/3 \rceil - 1}{\log 3 - 1} + 1 \le L \le \frac{3(\log\lceil n/3 \rceil - 1)}{\log 3} + 1$$
- liczba bitów wyniku – $\log_2 n$

- zliczanie zer – liczba jedynek w słowie zanegowanym jest równa liczbie zer w słowie oryginalnym

Alternatywne konstrukcje wielopoziomowego drzewa CSA*)

od góry (*top-down*)

liczba operandów	*struktura*
$N=N_0$	$(N_0/3)*(3,2)$
$(N_0/3)*2+\|N_0\|_3=N_1$	$(N_1/3)*(3,2)$
… …	… …
(6/3)∗2+0=4	1+1∗(3,2)
(4/3)∗2+1=3	1∗(3,2)

L
$L-1$
…
2
1

od dołu (*bottom-up*) – drzewo Wallace'a

liczba operandów	*struktura*
$k_{L-1}<N<k_L$	$(N-k_{L-1})*(3,2)+$
$k_{L-2}+\lfloor k_{L-2}/2\rfloor=k_{L-1}$	$\lfloor k_{L-1}/3\rfloor*(3,2)+\|k_{L-1}\|_3$
… …	… …
$3+\lfloor 3/2\rfloor=4=k_2$	$\lfloor 3/2\rfloor*(3,2)+1$
$2+\lfloor 3/2\rfloor=3=k_1$	$\lfloor 2/2\rfloor*(3,2)$

redukcja od poziomu L,
łatwiejsza konstrukcja drzewa

kumulacja operandów od poziomu 1
$k_{L+1}=k_L+\lfloor k_L/2\rfloor$, $k_0=2$

L	*operandy*	*struktura*		*operandy*	*struktura*
7	21	7∗(3,2)		27	9∗(3,2)
6	7∗2=14	2+4∗(3,2)		9∗2=18	6∗(3,2)
5	4∗2+2=10	1+3∗(3,2)		6∗2=12	4∗(3,2)
4	3∗2+1=7	1+2∗(3,2)		4∗2=8	2+2∗(3,2)
3	2∗2+1=5	2+1∗(3,2)		2∗2+2=6	2∗(3,2)
2	1∗2+2=4	1+1∗(3,2)		2∗2=4	1+1∗(3,2)
1	1∗2+1=3	1∗(3,2)		1∗2+1=3	1∗(3,2)

operandy	*struktura*
20…27	1…8∗(3,2)+…
8∗2+3=19	6∗(3,2)+1
6∗2+1=13	4∗(3,2)+1
4∗2+1=9	3∗(3,2)
2∗3=6	2∗(3,2)
2∗2=4	1∗(3,2)+1
1∗2+1=3	1∗(3,2)

Konwertery (*k, m*) w systemach dwójkowych*

$$\beta=2 \Rightarrow k \le 2^m - 1,\ m = \lceil \log_2(k+1) \rceil$$

- elementarny reduktor (3,2) – jednopozycyjny sumator binarny (FA)
- *konwerter* (*k,m*) (*licznik jedynek*)
  - – koduje liczbę jedynek z *k* wejść na *m* wyjściach
  - – drzewo (3,2) lub projekt indywidualny, np. licznik (4,3)

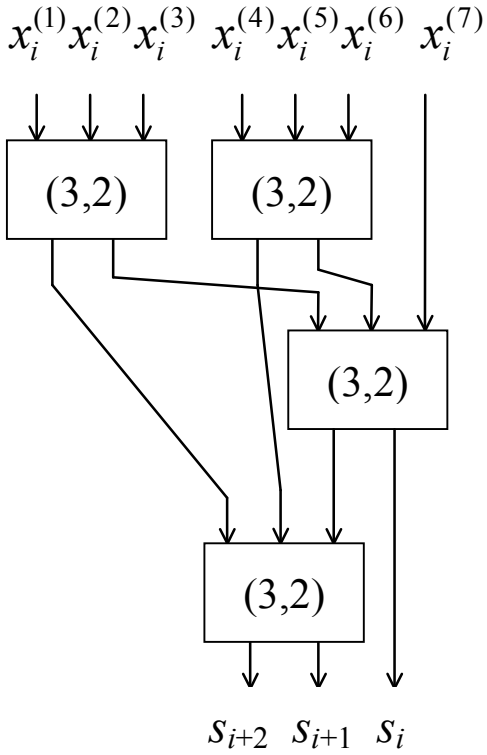
$$u^{(0)} = (x \oplus y) \oplus (z \oplus v)$$

$$u^{(1)} = (x \oplus y)(z+v) + (y \oplus z)(x+v) + (z \oplus v)(x+y)$$

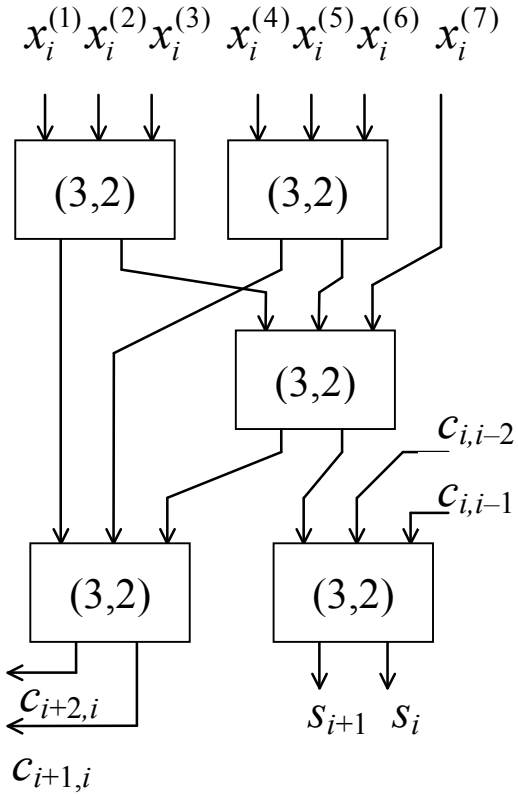
$$u^{(2)} = xyzv$$

- *reduktor* (*k*,2) – koduje liczbę jedynek z *k* wejść na 2 wyjściach sumy i pewnej liczbie wyjść przeniesień (kumulacja przeniesień)

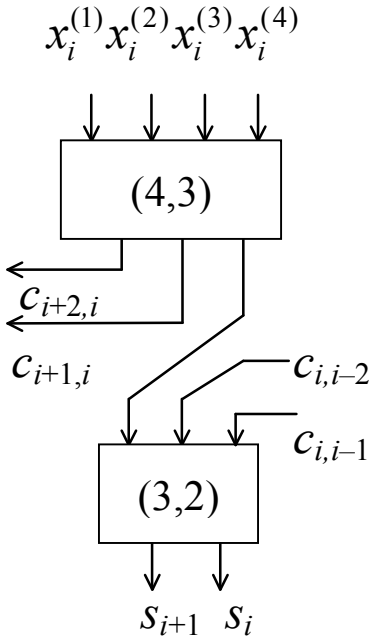
Konwertery (*k*, *m*) i reduktory (*k*, 2)*



Konwerter (7,3)



Reduktor (7,2)



Reduktor (4,2)

Reduktory wielokolumnowe $(k_{s-1},\ldots,k_1,k_0,m)$*

Dodawanie operandów o rosnących wagach (k_i o wadze β^i, i=0,1,…,s)
suma na m pozycjach – wektor o l składowych:
– jednooperandowa s-pozycyjna suma o wadze 2^0,
– wielooperandowe przeniesienie wektorowe o wagach operandów $(2^s)^i$

Warunek zakodowania wyniku na m pozycjach

$$\beta^m - 1 \geq \sum_{i=0}^{s-1} k_i(\beta-1)\beta^i$$

w systemie dwójkowym $\quad 2^m - 1 \geq \sum_{i=0}^{s-1} k_i 2^i$

warunek realizowalności licznika (k, k, ..., k, m)

$$2^m - 1 \geq k(2^s - 1)$$

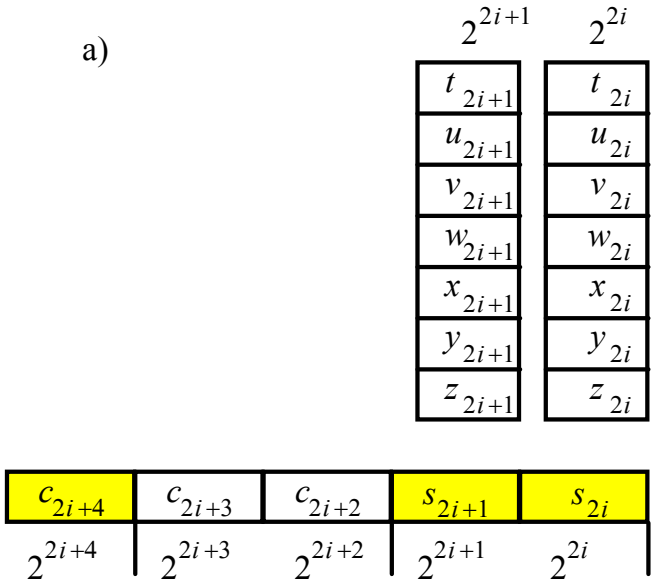
$m \leq 2s \Rightarrow$ suma k operandów s-pozycyjnych jest najwyżej 2-operandowa

$$2^{2s} - 1 \geq 2^m - 1 \geq k(2^s - 1) \Rightarrow k \leq 2^s + 1$$

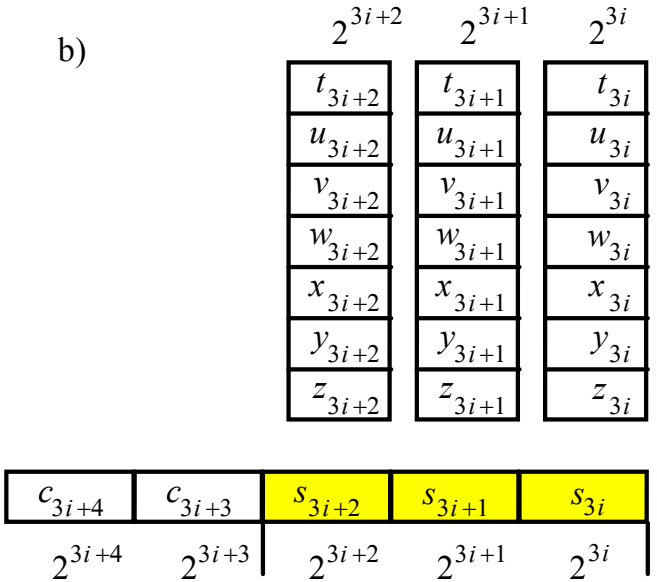
Parametry optymalnych reduktorów *s*-kolumnowych*

s	1	2	3	4	5	6	7
$m=2s$	2	4	6	8	10	12	14
k	3	5	9	17	33	65	129

a)



b)



Schemat dodawania w układach: a) (7,7,5), b) (7,7,7,5)

UKŁADY MNOŻĄCE

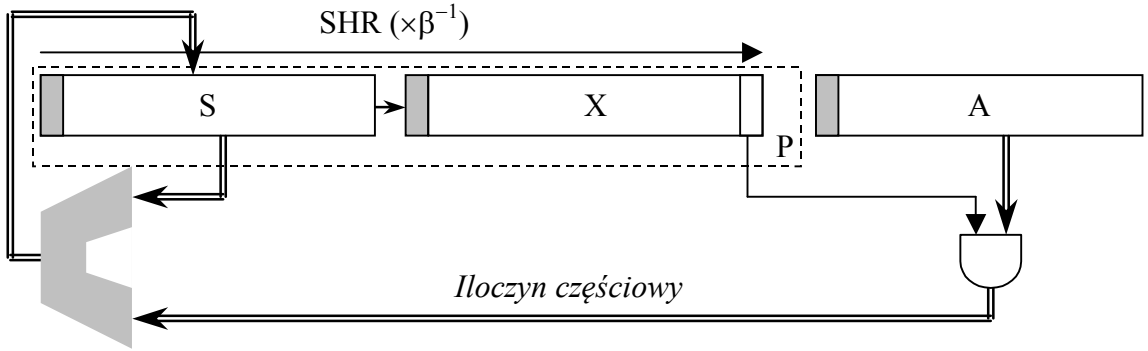
Podstawowy schemat mnożenia (dodaj-przesuń)

Krok 1. C||S/P ← (S/P) + x_i(A)

Krok 2. C||S/P||X/P ← 2^{-1}(C||S/P||X/P) = ShR(C||S/P||X/P)

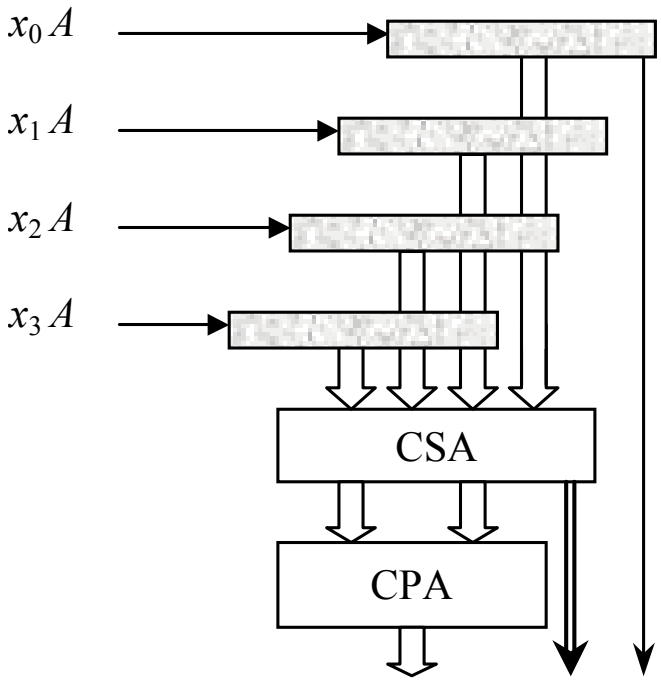
Krok 3. i=i+1. Jeśli $i < k+m$, wróć do kroku 1.

Wynik: $A \cdot X$=(S/P||X/P)

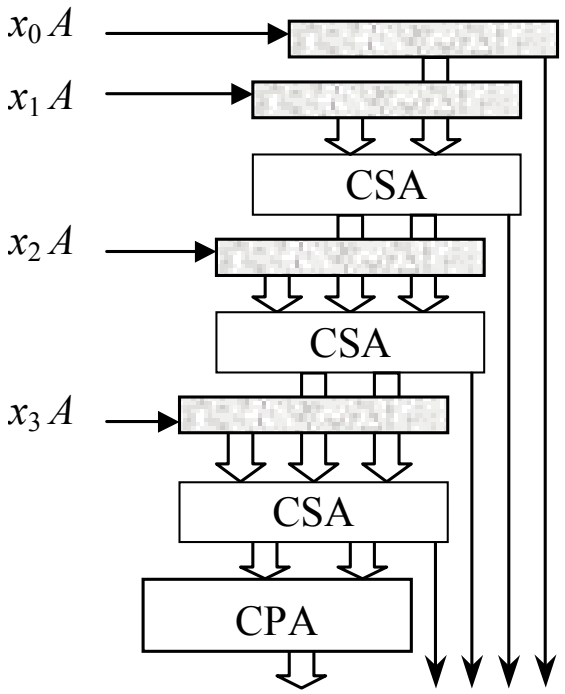


Schemat blokowy układu mnożącego: S/P – rejestr sum częściowych, X/P – rejestr mnożnika, A – rejestr mnożnej, C – rejestr przeniesienia

Schematy szybkiego mnożenia



akumulacja równoległa



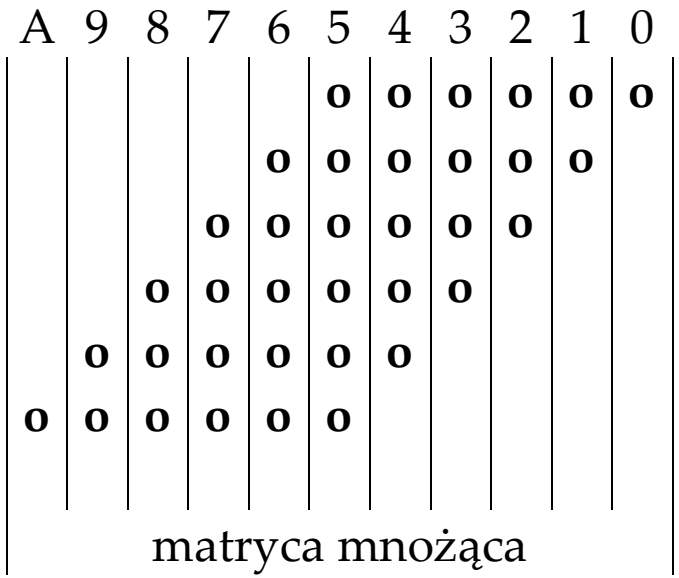
akumulacja sekwencyjna

- akumulacja równoległa – drzewiasta struktura CSA,
- akumulacja sekwencyjna – liniowa struktura CSA, matryca mnożąca

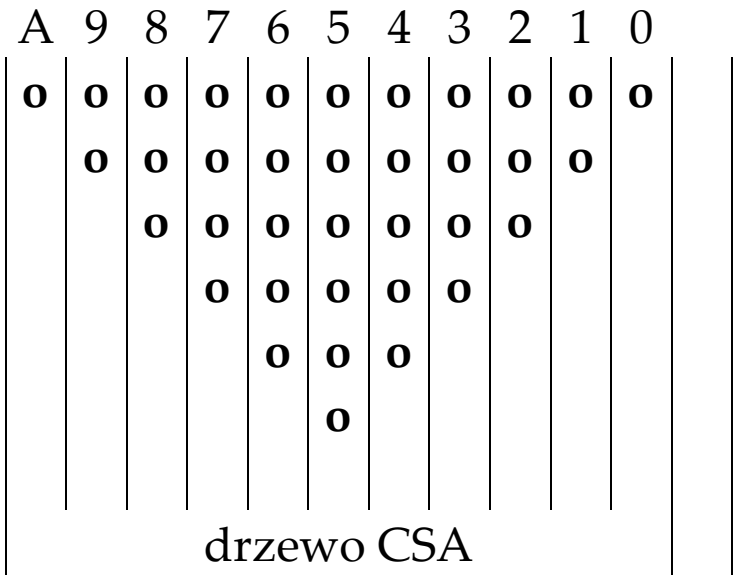
Akumulacja iloczynów częściowych

- sumatory wielooperandowe CSA

różne wagi iloczynów częściowych



różna liczba operandów jednej wagi

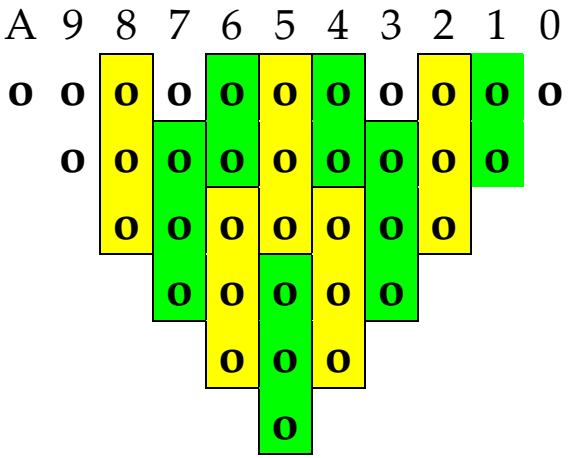


drzewo CSA

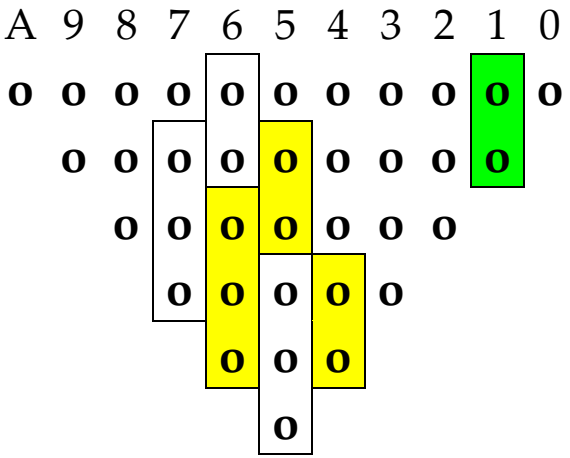
- szybka redukcja operandów w najdłuższej kolumnie
- redukcja do 1 operandów najniższych wag (krótsze końcowe dodawanie)

Optymalizacja struktury CSA (1)

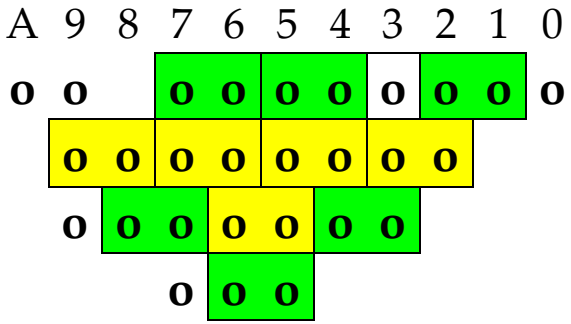
redukcja maksymalna

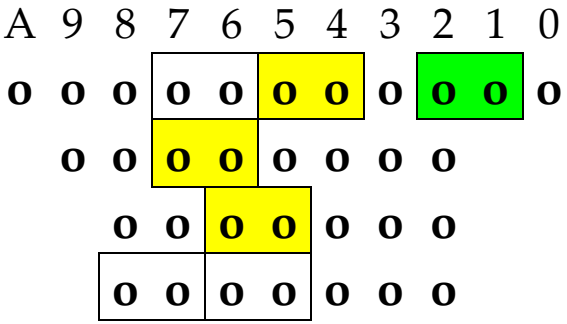


drzewo Wallace'a



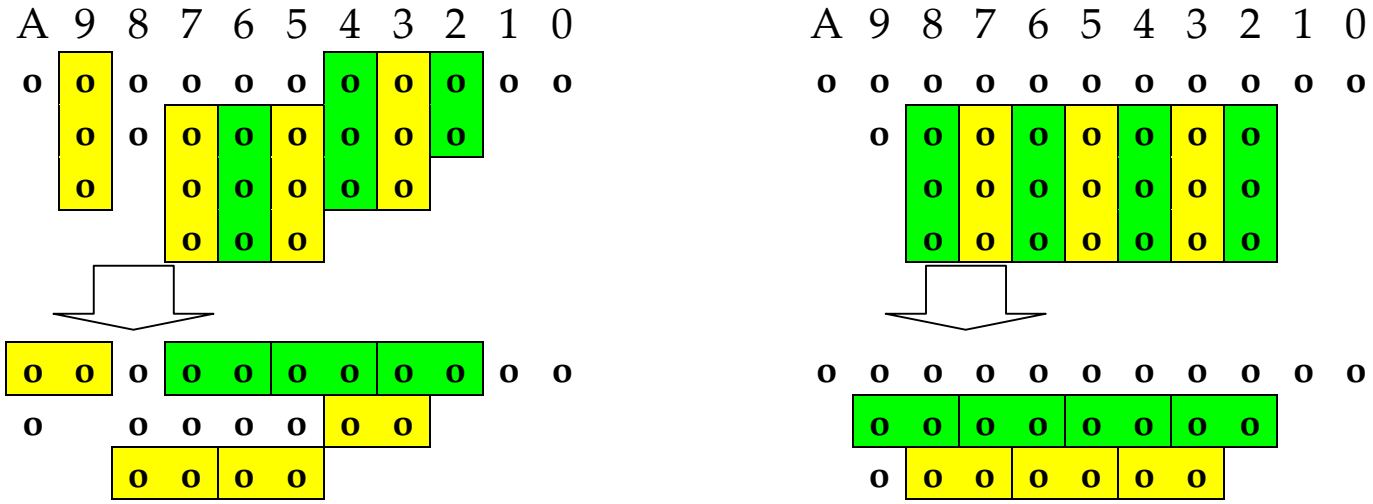
CSA, poziom 3 – wejścia układów (3,2) lub (2,2)



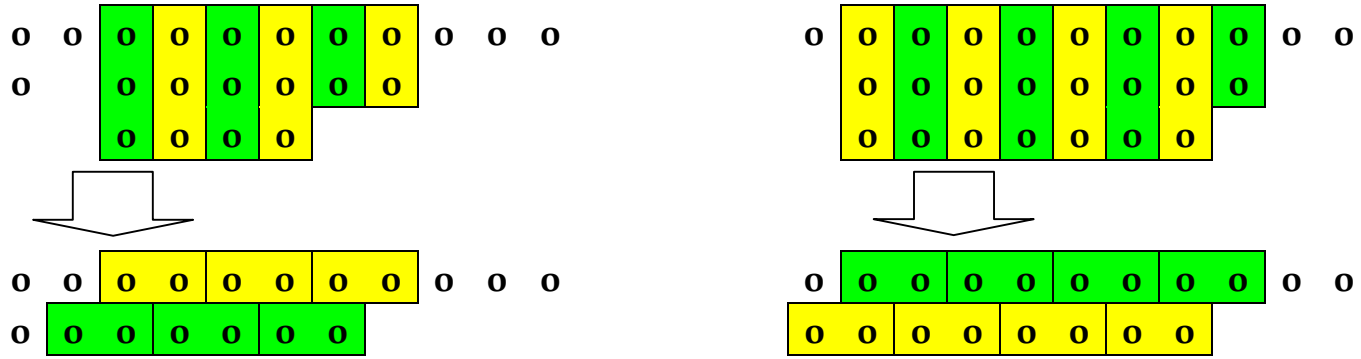


CSA, poziom 3 – wynik redukcji: wyjścia układów (3,2) lub (2,2)

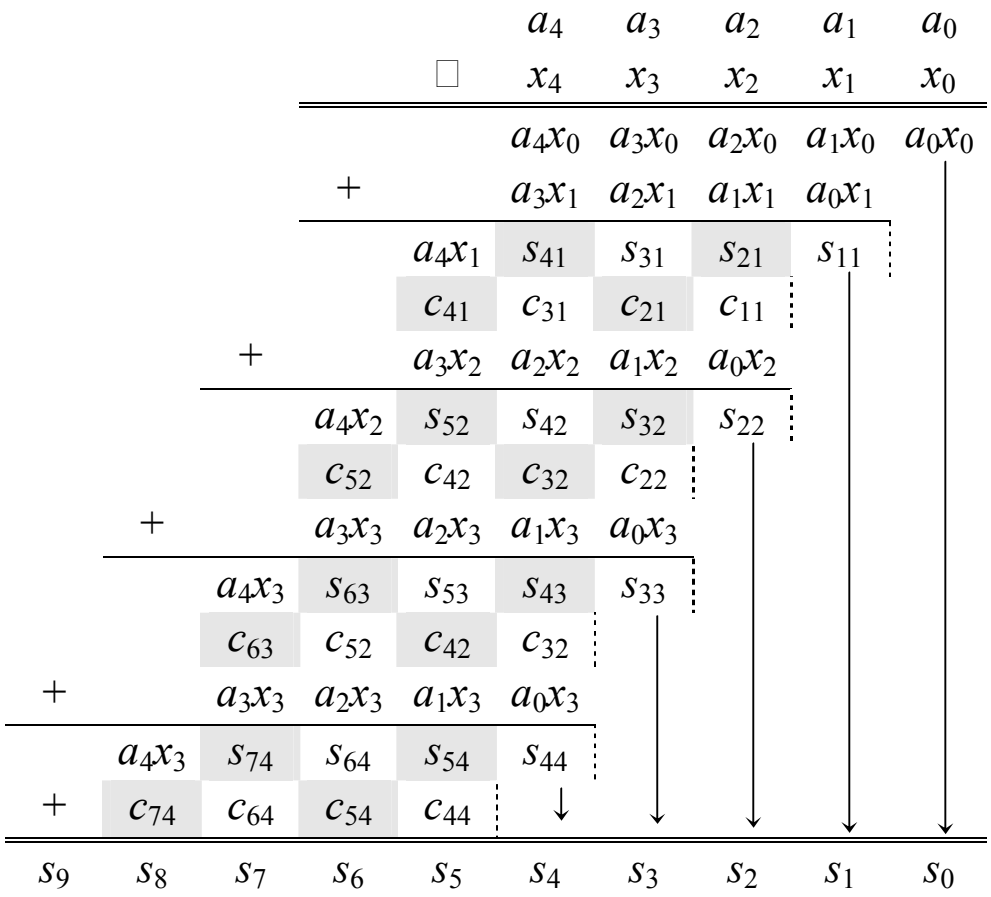
Optymalizacja struktury CSA (1)



CSA, poziom 2

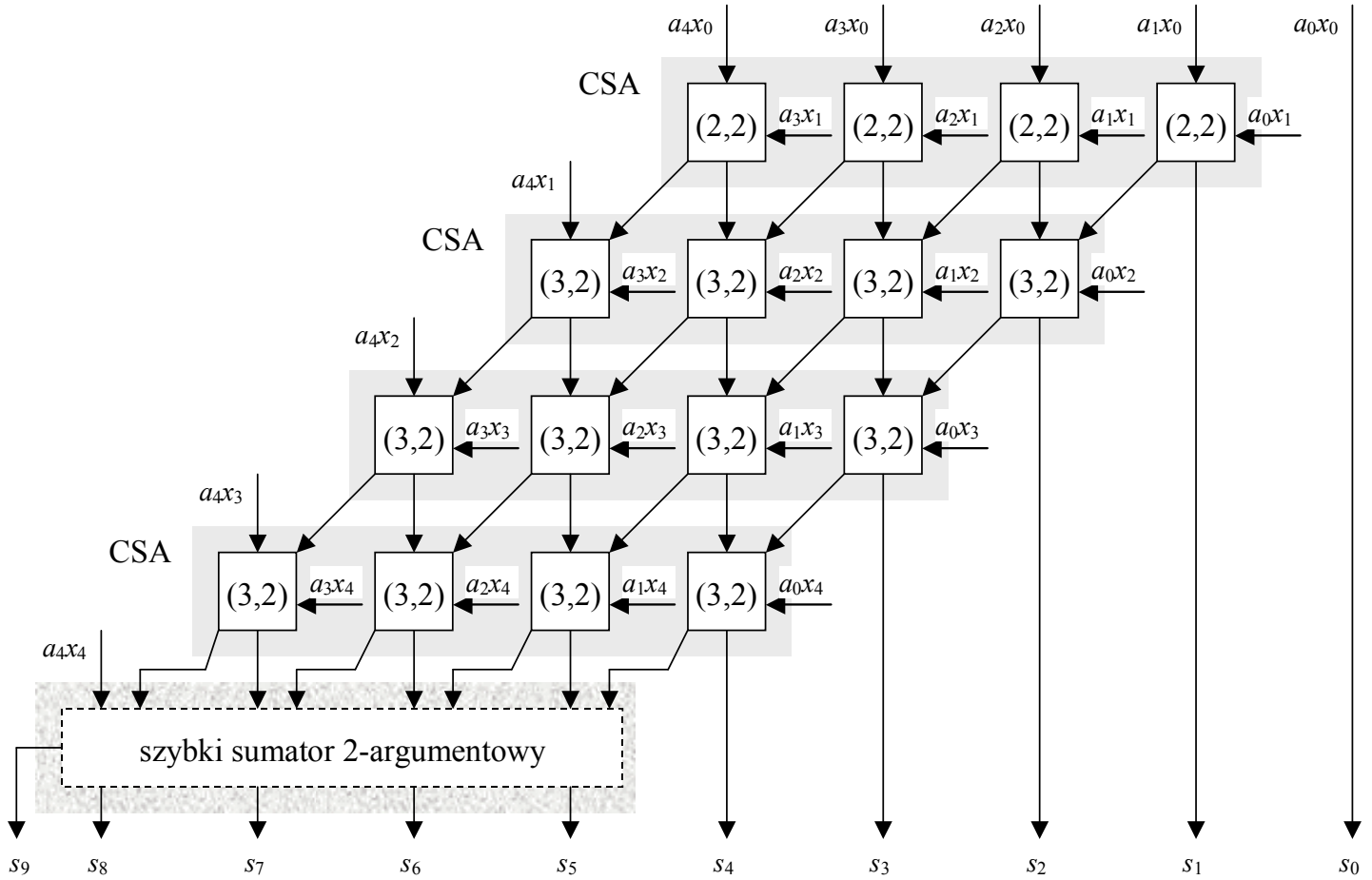


Matrycowe układy mnożące – schemat mnożenia

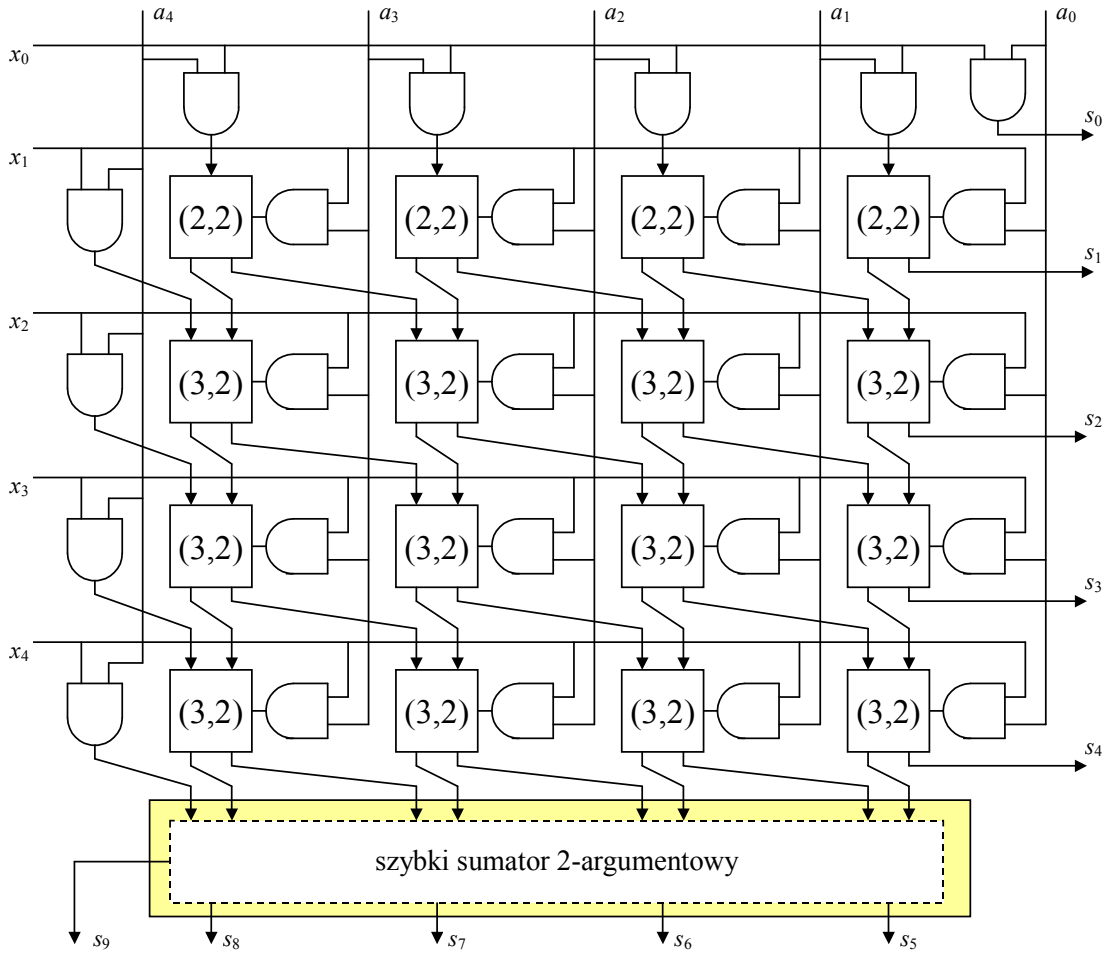


s_{ji} oraz c_{ji} – sumy i przeniesienia na pozycji j w i-tym kroku akumulacji

Matryca mnożąca kodu naturalnego (Brauna)

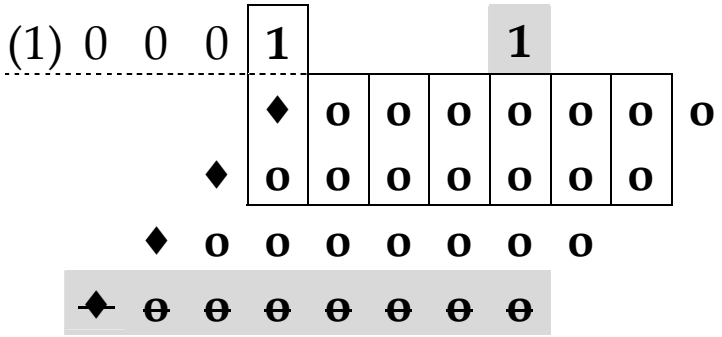


Multiplikator Brauna (Braun multiplier)



Konstrukcja matrycy mnożącej kodu uzupełnieniowego

realizacja algorytmu mnożenia „bez rozszerzeń”:



(♦– negacja najbardziej znaczącego bitu operandu, o – negacja bitu)

dodawanie stałych korygujących 2^{k-1} i -2^{k+m-1} oraz 2^{m-1} (uzupełnianie mnożnej):

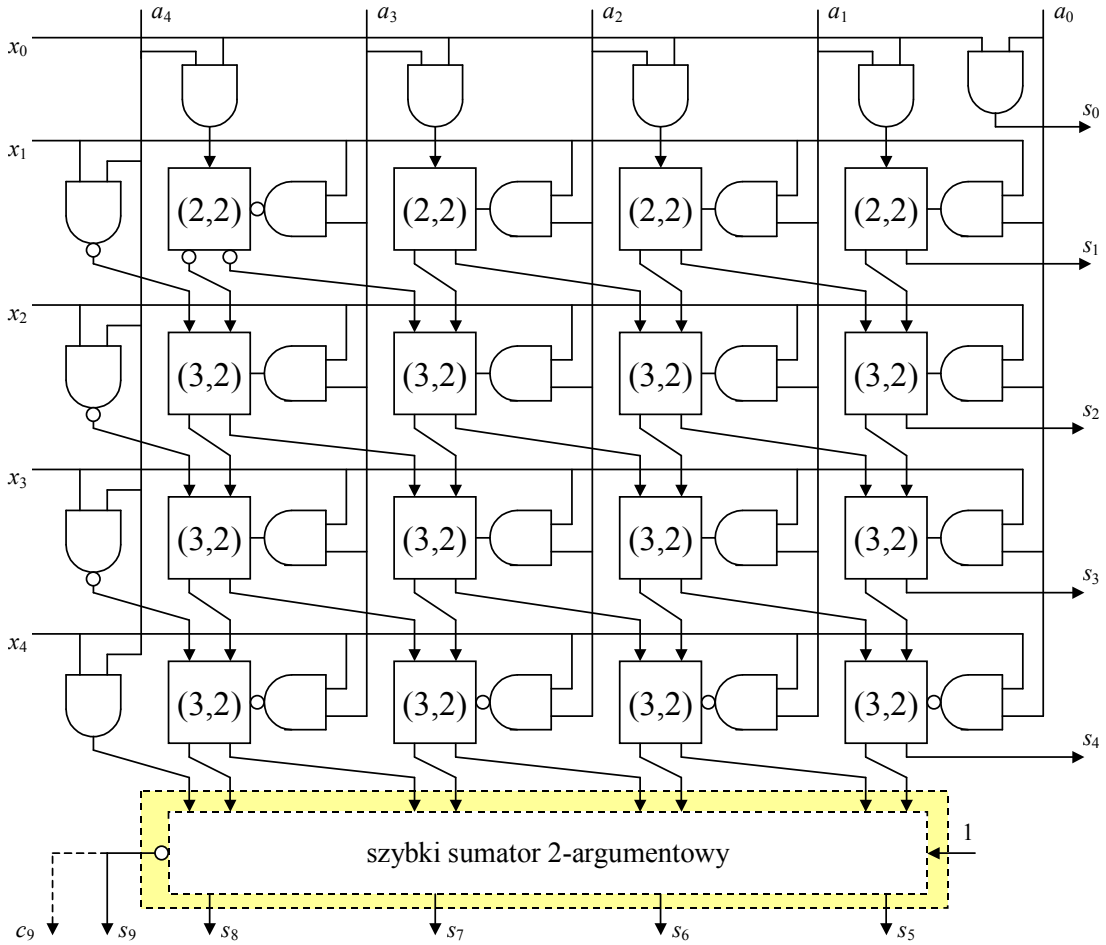
- korygująca „1” na pozycji k–1 (2^{k-1}) $\Rightarrow$ $s+1=2c_+ + s^*$ $\Rightarrow$ $s^*=1\oplus s$, $c_+=s$
- dodanie 2^{m-1} – modyfikacja półsumatora pozycji m–1 w pierwszej linii

$$x+y+1=2c_+ + s \;\Rightarrow\; s=\overline{x\oplus y},\; c_+ = x+y \;\text{ lub }\; s=\overline{\bar{x}\oplus\bar{y}},\; c_+=\overline{\bar{x}\cdot\bar{y}}$$

- dodanie -2^{n+l-1} – korekcja przeniesienia z najwyższej pozycji iloczynu , zgodnie z zależnością $c_-+1=2c_+ + s$, czyli $c_+=c_-$ oraz $s=1\oplus c_-$

- *matryca kwadratowa* ($k=m$) – ($2^{k-1}+2^{k-1}=2^k$) $\Rightarrow$ jedna korekcja na pozycji k

Matryca mnożąca kodu uzupełnieniowego (Baugh'a-Wooley'a)



(ostatni iloczyn częściowy: negacja bitów mnożnej i korekcja)

Alternatywny układ mnożący kodu U2

iloczyny częściowe lub iloczyny elementarne mogą być liczbami ujemnymi

$$\left(-a_{k-1}2^{k-1}+\sum_{i=0}^{k-2}a_i 2^i\right)\cdot\left(-x_{m-1}2^{m-1}+\sum_{j=0}^{m-2}x_j 2^j\right)=$$

$$=x_{m-1}a_{k-1}2^{m+k-2}+\sum_{j=0}^{m-2}\sum_{i=0}^{k-2}x_j a_i 2^{i+j}+\left(-a_{k-1}2^{k-1}\sum_{j=0}^{m-2}x_j 2^j\right)+\left(-x_{m-1}2^{m-1}\sum_{i=0}^{k-2}a_i 2^i\right).$$

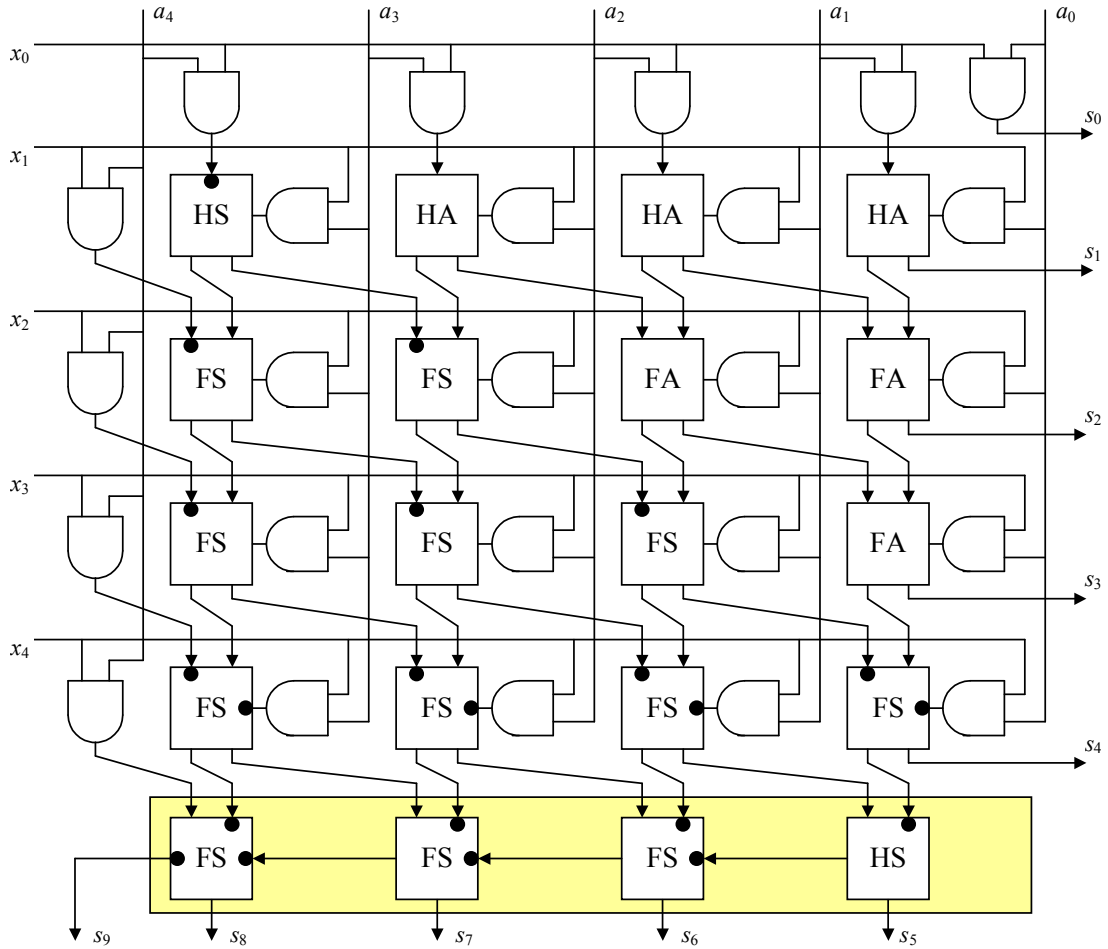
- wagi operandów (1-bitowych iloczynów) mogą być ujemne
  → wystarczy zmienić znaki wag wejść i wyjść niektórych sumatorów

- zastąpienie sumatorów FA ((3,2)) realizujących dodawanie $x+y+z=2c+s$
  układami odejmującymi FS ($x-y-z=-2c+s$) lub FS^D ($x+y-z=2c-s$)

- struktura logiczna FS i FS^D identyczna
- przeciwne wagi wejść i wyjść, bo

$$x-y-z=-(z+y-x) \quad \text{oraz} \quad -(2c-s)=-2c+s$$

Alternatywna matryca mnożąca kodu uzupełnieniowego



(• – wejścia o ujemnej wadze)

Charakterystyki matryc mnożących

złożoność (mnożnik m-bitowy, mnożna k-bitowa)

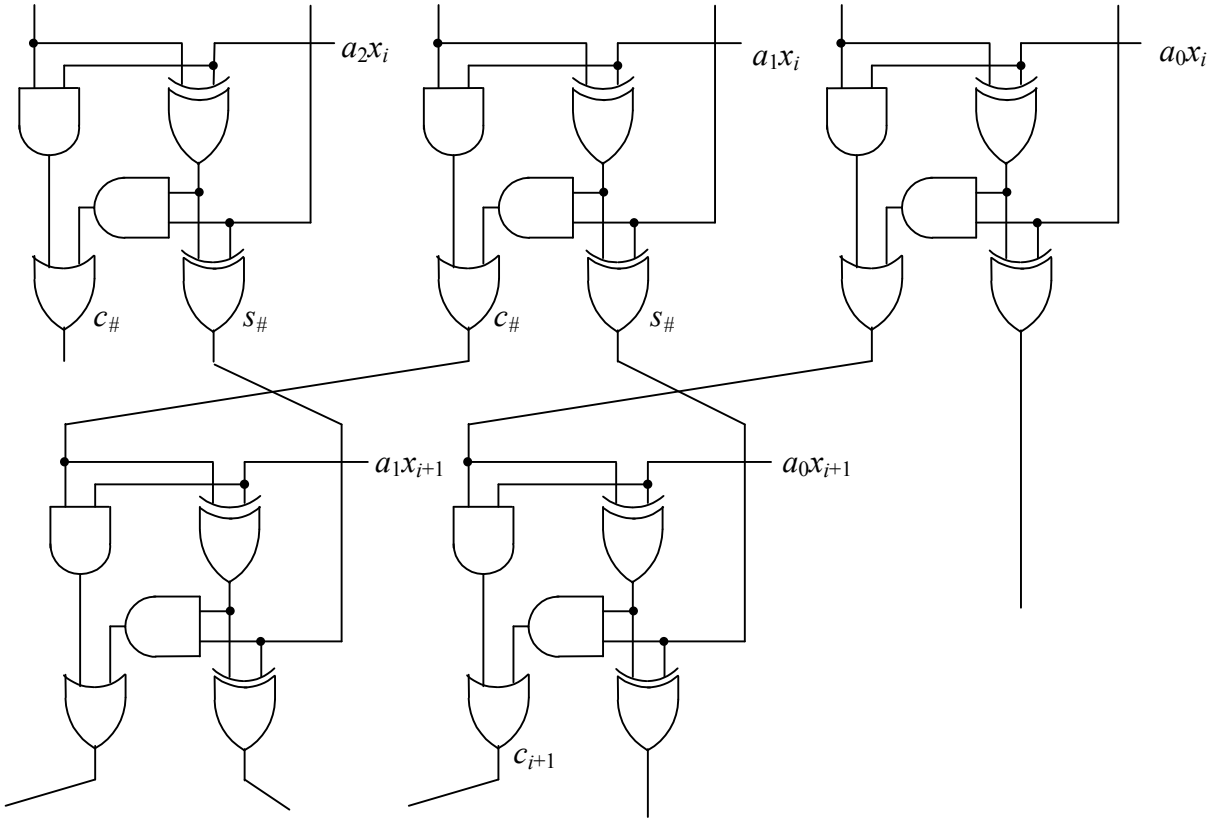
- $A = 8(m-1)k$ (dodatkowa bramka AND na każdy akumulowany bit)
- $T = 3(m-1) + T_{CPA(k)} \geq 3m + 2\log k - 1$
  (odpowiednie łączenie poziomów daje opóźnienie 6 na dwóch poziomach)

podatność na przetwarzanie potokowe (*pipelining*)

- dla danej pary operandów w danej chwili jest wykonywane dodawanie tylko na jednym poziomie układu matrycowego,
- na innych poziomach można *w tym samym czasie* wykonać wcześniejsze lub późniejsze fazy mnożenia innych par operandów
- niezbędne rozbudowanie o dodatkowe układy transmitujące wyniki dodawania na mniej znaczących pozycjach oraz układ synchronizacji.

- przepustowość układu zależy od szybkości końcowego dodawania
  w seryjnym mnożeniu końcowy CPA jako kaskada CSA
  szybkość bliska szybkości dodawania 1-bitowego!

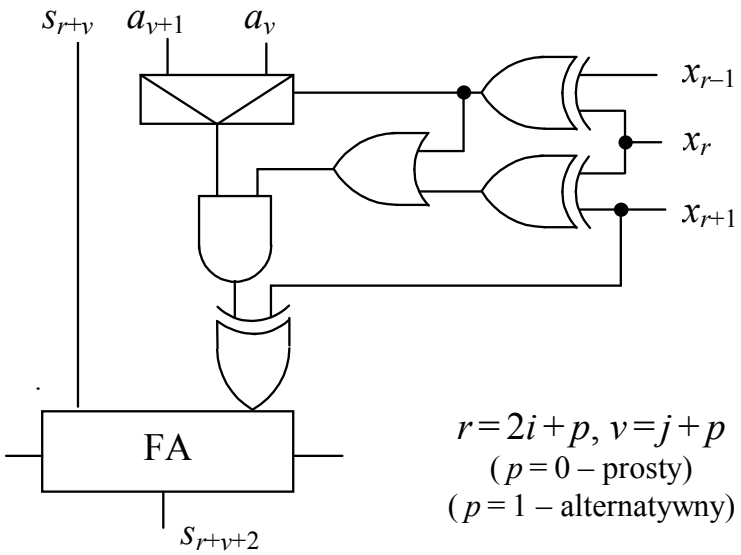
Optymalne łączenie poziomów CSA w matrycy mnożącej*



maksymalne opóźnienie przez 2 poziomy – (2+4) lub (4+2), czyli zawsze 6

Realizacja przekodowania Bootha-McSorleya w matrycy*

- możliwe zastosowanie algorytmu Bootha/McSorleya



- „brak podwojenia” = $x_r \oplus x_{r-1}$,
- „odejmowanie” = x_{r+1},
- „brak zerowania” = $(x_r \oplus x_{r+1}) + (x_r \oplus x_{r-1})$,

Strukturalizacja układów mnożących

- układ mnożący $kn \times kn$ – złożenie układów mnożących $n \times n$:

$$AX = \left(\sum_{s=0}^{k-1} A_s 2^{sn} \right) \left(\sum_{s=0}^{k-1} X_s 2^{sn} \right) = \sum_{j=0}^{k-1} 2^{jn} \sum_{i=0}^{j} A_i X_{j-i} + \sum_{j=k}^{2k-2} 2^{jn} \sum_{i=j-k+1}^{k-1} A_i X_{j-i},$$

albo w postaci skróconej

$$AX = \sum_{j=0}^{2k-2} 2^{jn} \sum_{i=\max(0,j-k+1)}^{\min(j,k-1)} A_i X_{j-i}$$

gdzie $A_i = \sum_{j=0}^{n-1} a_{ni+j} 2^j$, $X_i = \sum_{j=0}^{n-1} x_{ni+j} 2^j$.

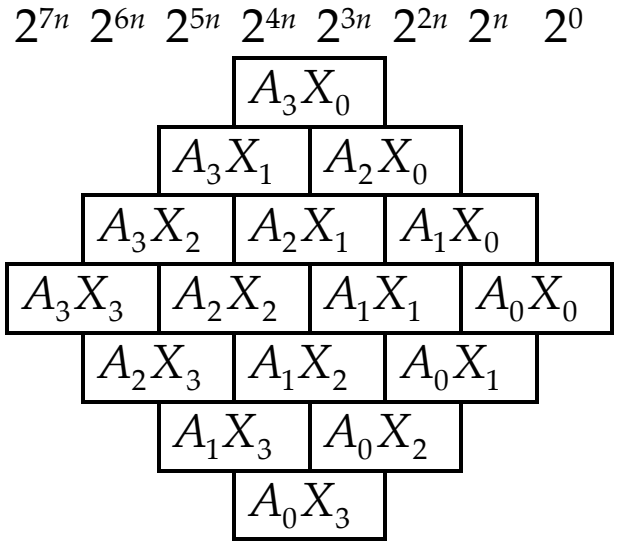
wyrównywanie (*alignment*)

- każdy $2n$-pozycyjny iloczyn $A_i X_{s-i}$ ma wagę 2^{ns}

$$(\mathbf{AX})_s = [A_s X_0, A_{s-1} X_1, \ldots, A_1 X_{s-1}, A_0 X_s]$$

- efekt – akumulacja $2k-1$ wielooperandów różnego rozmiaru
  zamiast k^2 operandów o identycznej wielkości

Wyrównanie operandów



Wyrównanie operandów w układzie mnożącym $4n \times 4n$

- w kodzie U2 – niezbędne uwzględnienie rozszerzenia znakowego

efekt – liczba operandów w j-ej grupie wynosi $2j+1$ osiągając maksimum $4k-3$,
$\rightarrow$ niweczy to zysk wynikający ze strukturalizacji.

$\rightarrow$ przekonstruowanie sumatora wielooperandowego CSA.

Mnożenie wielokrotnej precyzji

W mnożeniu liczb rozszerzonej precyzji, $(A_{k-1}A_{k-2}...A_1A_0)\cdot(X_{k-1}X_{k-2}...X_1X_0)$:

- wszystkie iloczyny A_iX_j takie, że $i<k-1$, $j<k-1$, są dodatnie i mogą być realizowane jako mnożenie naturalne („mul" w architekturze IA-32)
- pozostałe iloczyny są iloczynami liczb znakowanych i muszą być realizowane jako mnożenie całkowite („imul" w architekturze IA-32).

Mnożenie długich liczb znakowanych (U2)

- najwyższe iloczyny (... $A_HX_\#$ oraz $A_\#X_H$)
  – mnożenie liczby dodatniej przez znakowaną !
  – dodawanie dodatniej i znakowanej !

Rozwiązanie 1:

- przekodowanie na dodatnie (podobnie jak w mnożeniu bez rozszerzeń)
- korekcja (podobnie jak w mnożeniu bez rozszerzeń)

Rozwiązanie 2:

- przekodowanie na wartości bezwzględne
- mnożenie dodatnich i wytworzenie znaku
- przekodowanie iloczynu na kod uzupełnieniowy