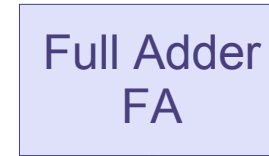
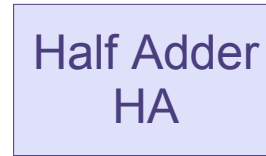




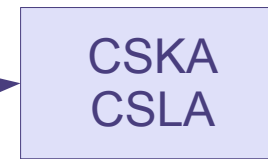
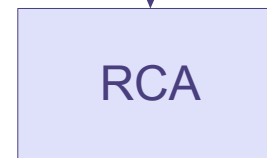
Sumatory

Architektury sumatorów (zarys)

Sumatory
1-bitowe

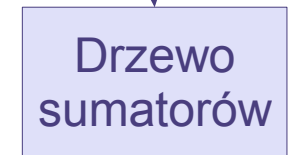
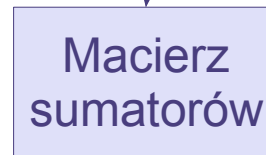


Sumatory z propagacją
Przeniesień – CPA
(Carry Propagate Adders)



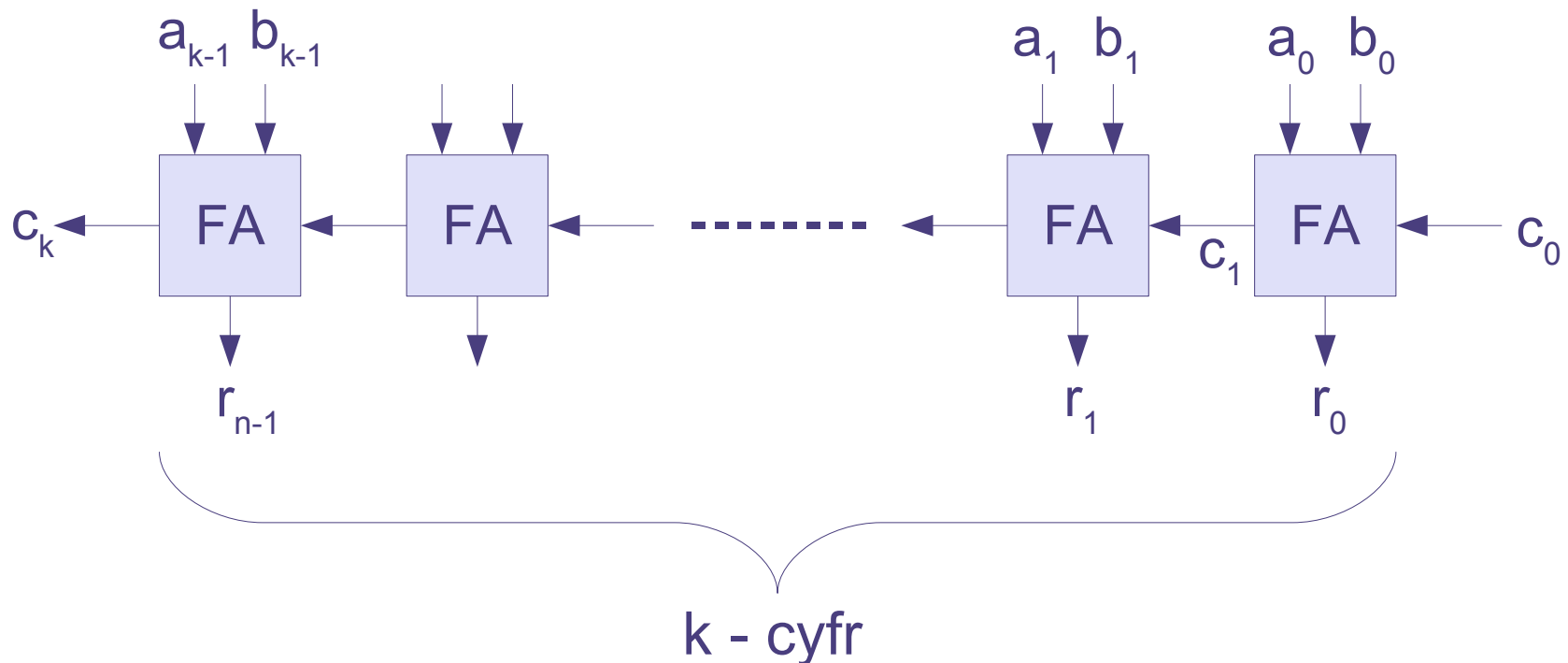
3-argumentowe

Sumatory
wieloargumentowe



Sumator z przeniesieniami szeregowymi

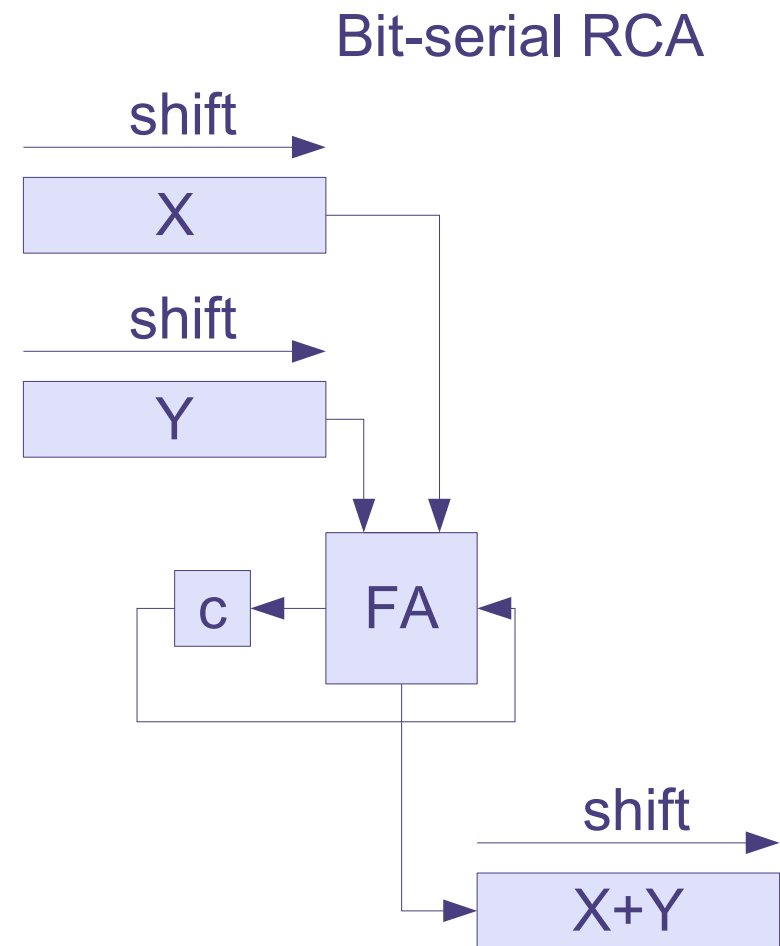
- RCA – Ripple Carry Adder (zwany też kaskadowym)
 - Najprostszym i najmniejszym, ale najwolniejszy
 - Szybkość (w najgorszym przypadku): $\Theta(k)$



RCA – implementacja bitowo-szeregowa

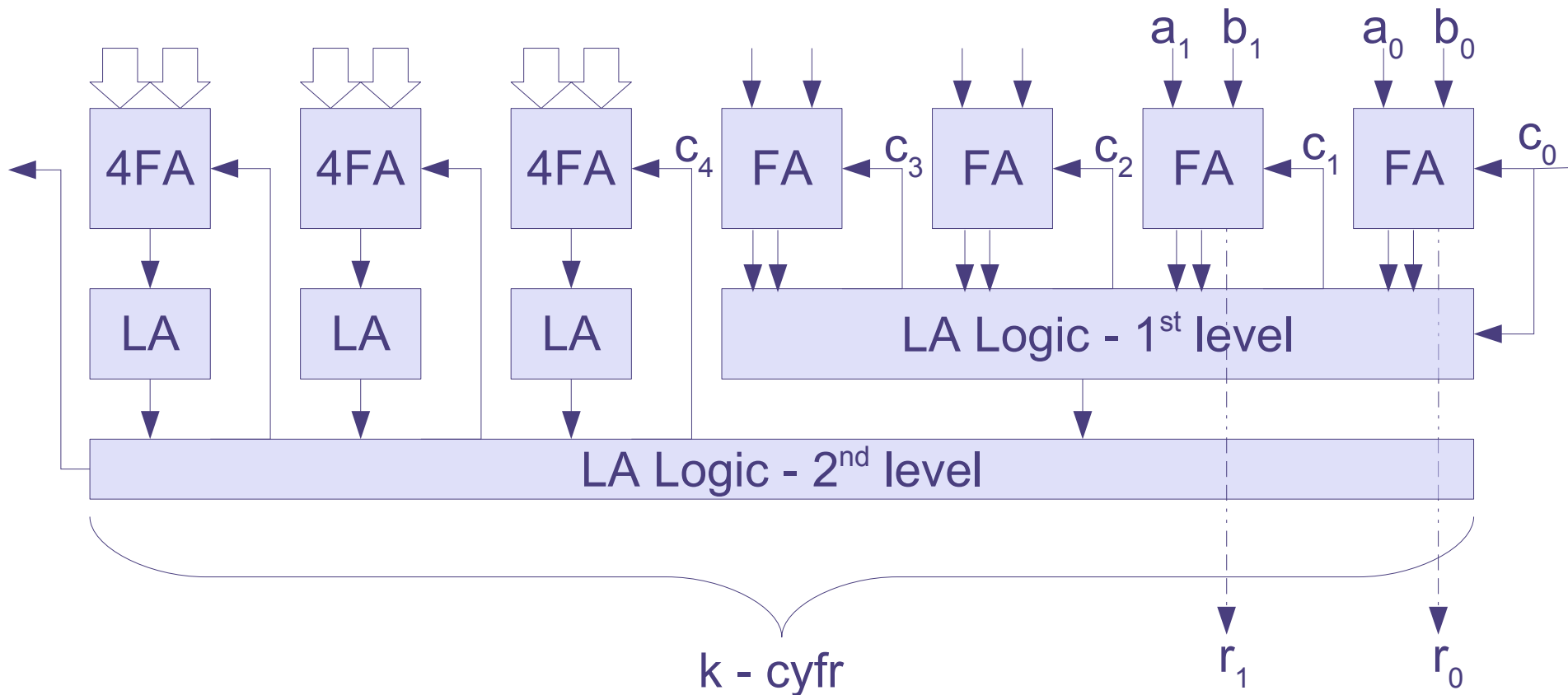
☉ Zalety (dla implementacji VLSI)

- ☉ mała liczba wyprowadzeń
- ☉ krótkie połączenia
- ☉ duża szybkość zegara
- ☉ mała powierzchnia
- ☉ niski pobór mocy
- ☉ Dobrze nadaje się do przetwarzania potokowego
- ☉ Szybkość bez zmian: $\Theta(k)$



Sumator z przeniesieniami równoległymi

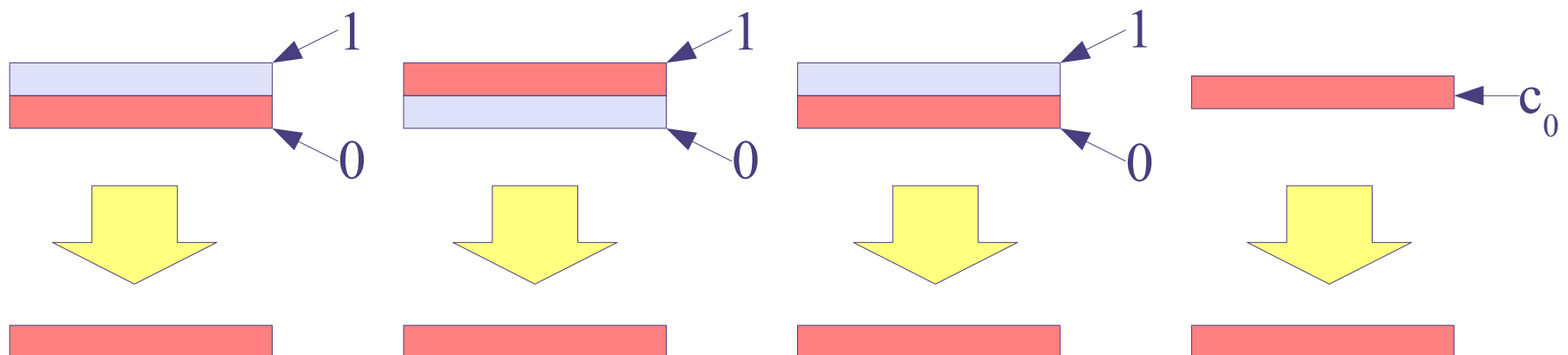
- CLA - Carry Lookahead Adder
 - Dodatkowe układy logiki obliczania przeniesień
 - Szybkość (w najgorszym przypadku): $\Theta(\log k)$



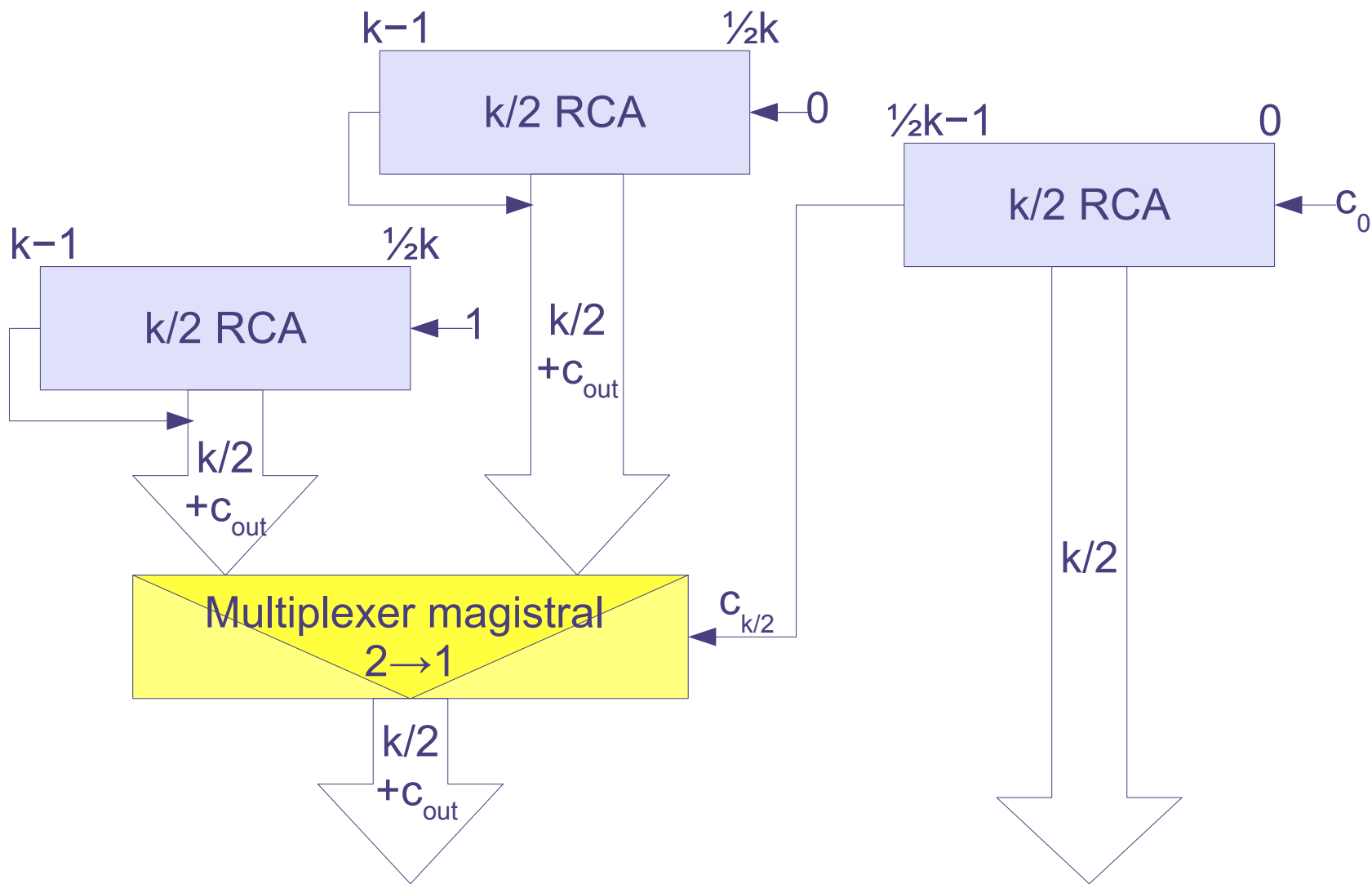
Sumator z wyborem przeniesienia

CSLA – Carry Select Adder

- Najstarsza architektura o szybkości $\Theta(\log k)$
- Dla kilku fragmentów argumentów obliczane są równoległe dwie wersje wyniku: dla przeniesień 0 i 1
- Końcowy wynik jest wybierany z gotowych alternatyw na podstawie obliczonych przeniesień
- Prosta koncepcja – skomplikowany sprzęt



Jednopoziomowy sumator CSLA

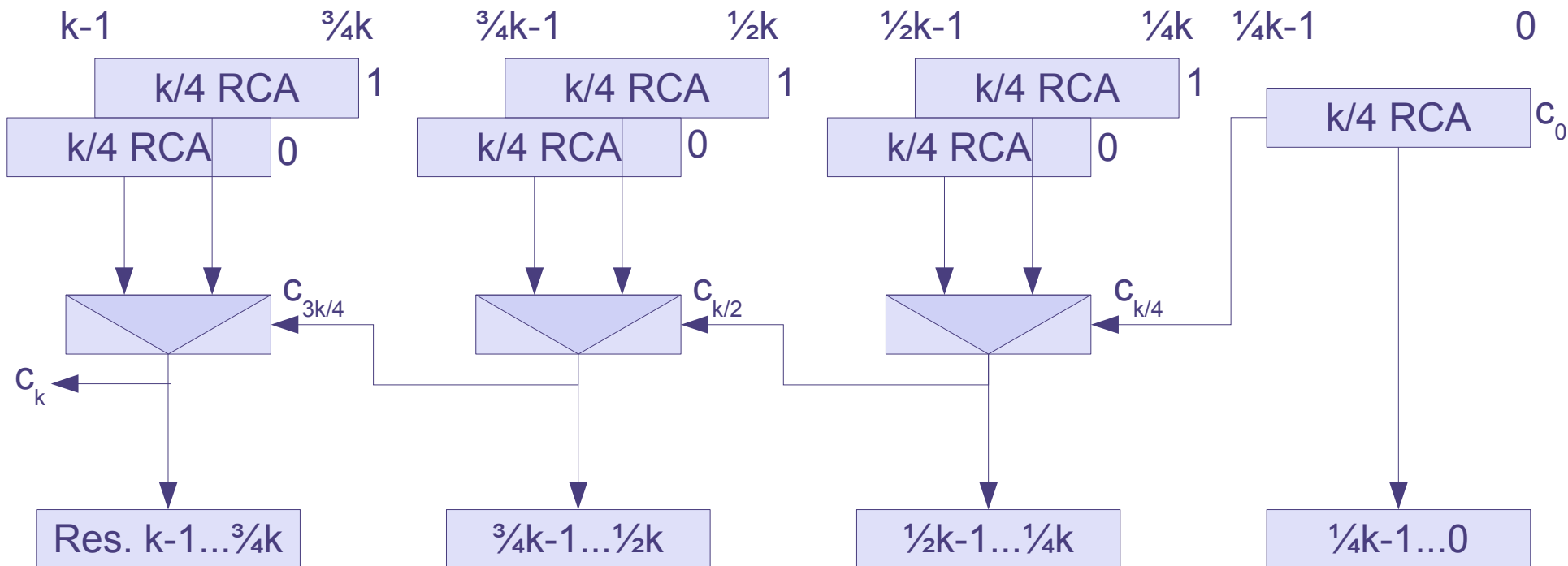


$c_{out} + k/2$ bardziej
znaczących bitów wyniku

$k/2$ mniej znaczących
bitów wyniku

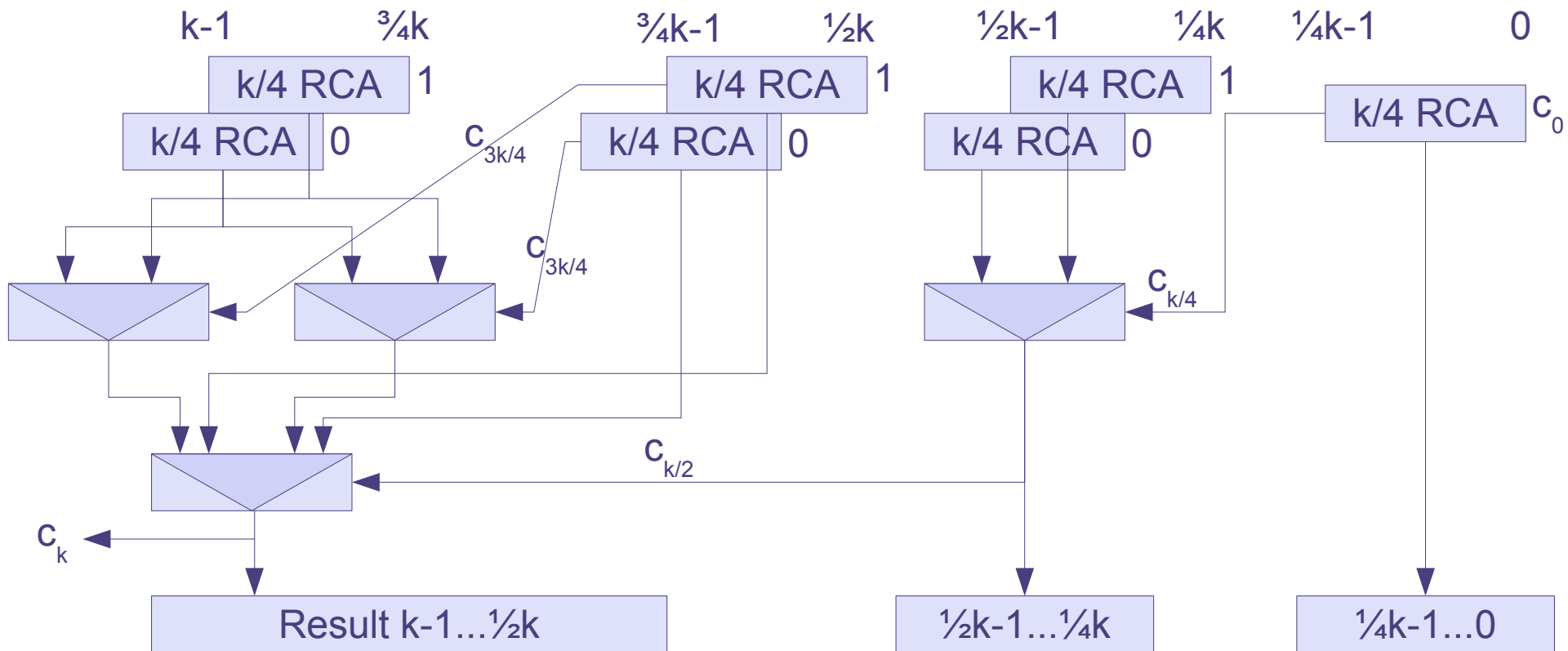
Propagacja w jednopoziomowym CSLA

- Architektura jednopoziomowa **nie** może być skalowana na dużą liczbę bloków ze względu na propagację sygnału pomiędzy blokami



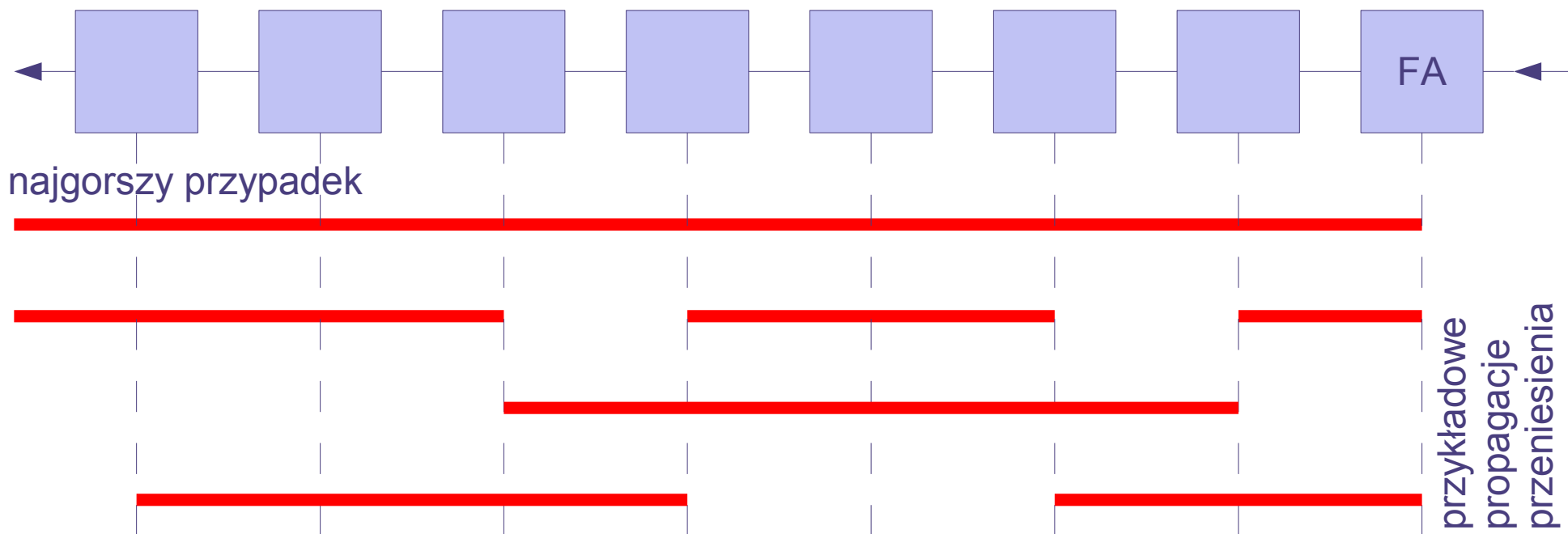
Dwupoziomowy sumator CSLA

- Wybór na drugim poziomie dotyczy dwukrotnie dłuższego fragmentu wyniku
- Koncepcja może być rozszerzana na większą liczbę poziomów, ale wzrasta złożoność sprzętu



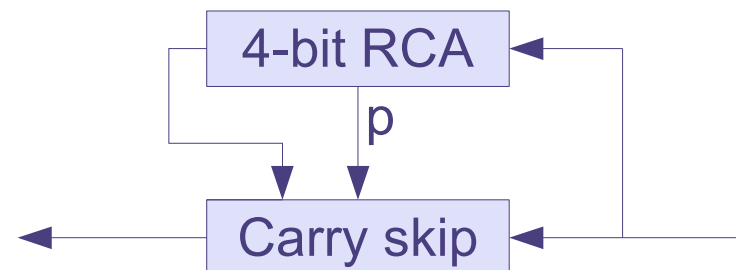
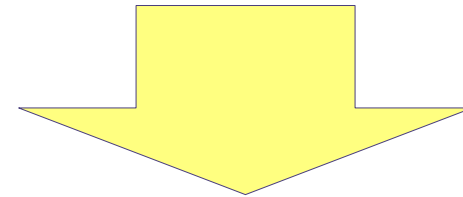
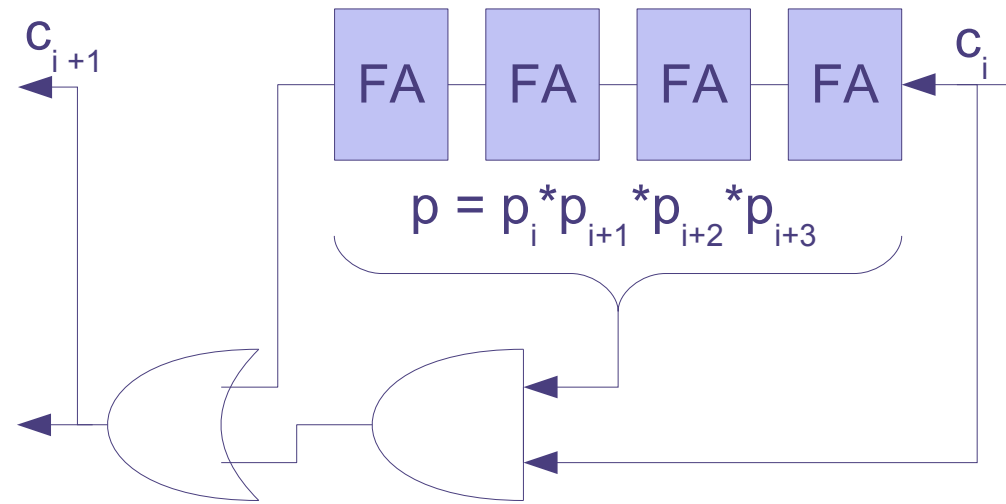
Sumator z przeskokiem przeniesień

- CSKA – Carry Skip Adder
- Przeniesienia mogą być propagowane ($p=1$), generowane ($g=1$) lub absorbowane ($p \cdot g=0$)
- Zezwolenie na przeskok przeniesienia ponad blokami z $p=1$ skraca ciągi propagacji przeniesienia



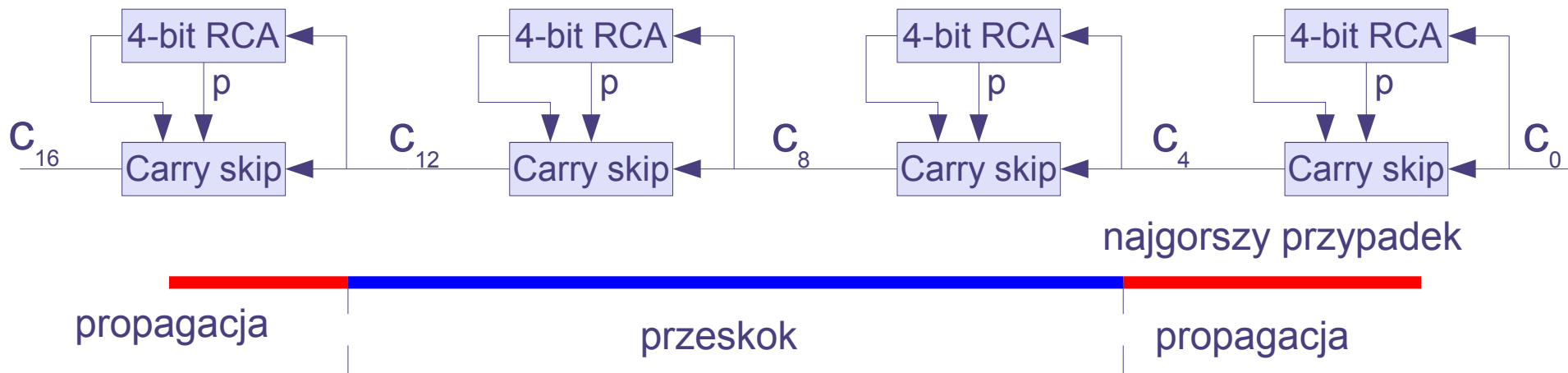
Blok przeskoku przeniesienia

- Przeniesienie jest przyspieszane dla grupy sumatorów z warunkiem $p=1$
- Obliczanie warunku przeniesienia jest proste: $p_i = a_i + b_i$



Analiza najgorszego przypadku CSKA

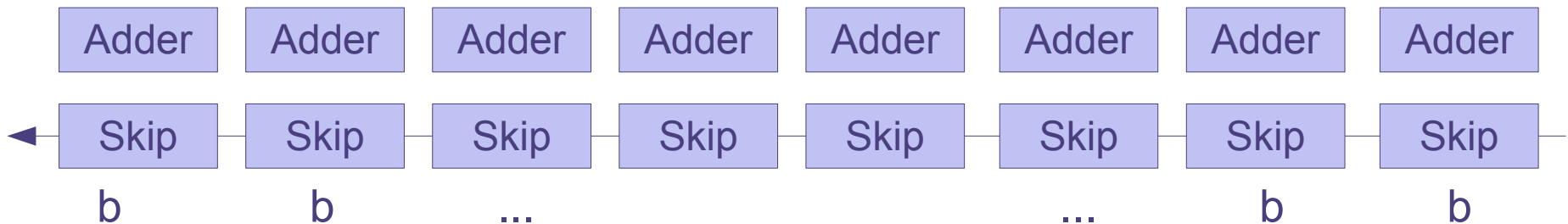
- 16-bitowy sumator CSKA
- Najdłuższa ścieżka propagacji przeniesienia:
 - propagacja przez bity 1-3 (i bramkę OR)
 - przeskok przez bity 4-11
 - propagacja przez bity 12-14



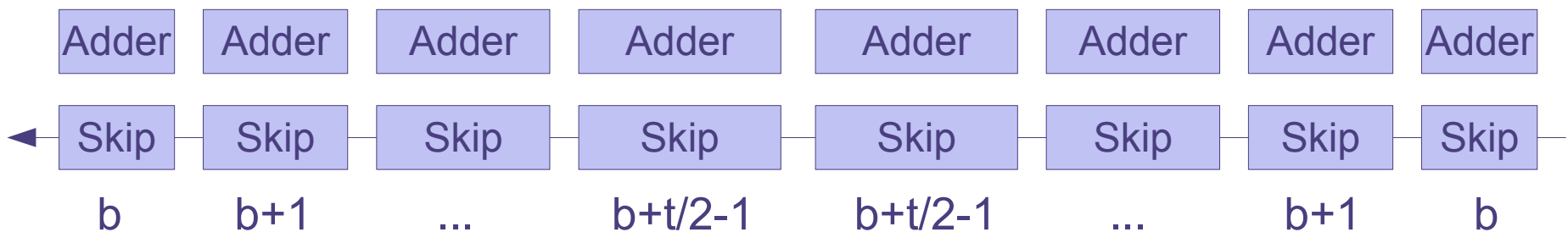
Architektura sumatora CSKA

☉ Bloki przeskoku przeniesień mogą mieć:

☉ stałą długość b



☉ zmienną długość (t – liczba bloków)



Wydajność sumatora CSKA

- Opóźnienie (T) ze względu na propagację dla sumatora CSKA o długości bloku b:
 - dla bloków o stałej długości:
 - $b_{opt} = (k/2)^{1/2}$
 - $T_{fixCSKA-opt} = 2(2k)^{1/2} - 3.5$
 - dla bloków o zmiennej długości:
 - $b_{opt} = 1$
 - $T_{varCSKA-opt} = 2k^{1/2} - 2.5$

Wydajność sumatora CSKA

Przykłady (przy założeniu jednostkowego opóźnienia dla FA):

- dla bloków o stałej długości:

- 16-bitowy sumator $\rightarrow b_{opt} \approx 3, T_{fixCSKA} \approx 8$

- 32-bitowy sumator $\rightarrow b_{opt} = 4, T_{fixCSKA} = 12.5$

- 64-bitowy sumator $\rightarrow b_{opt} \approx 6, T_{fixCSKA} \approx 19$

- dla t-bloków o zmiennej długości ($b_{opt} = 1$):

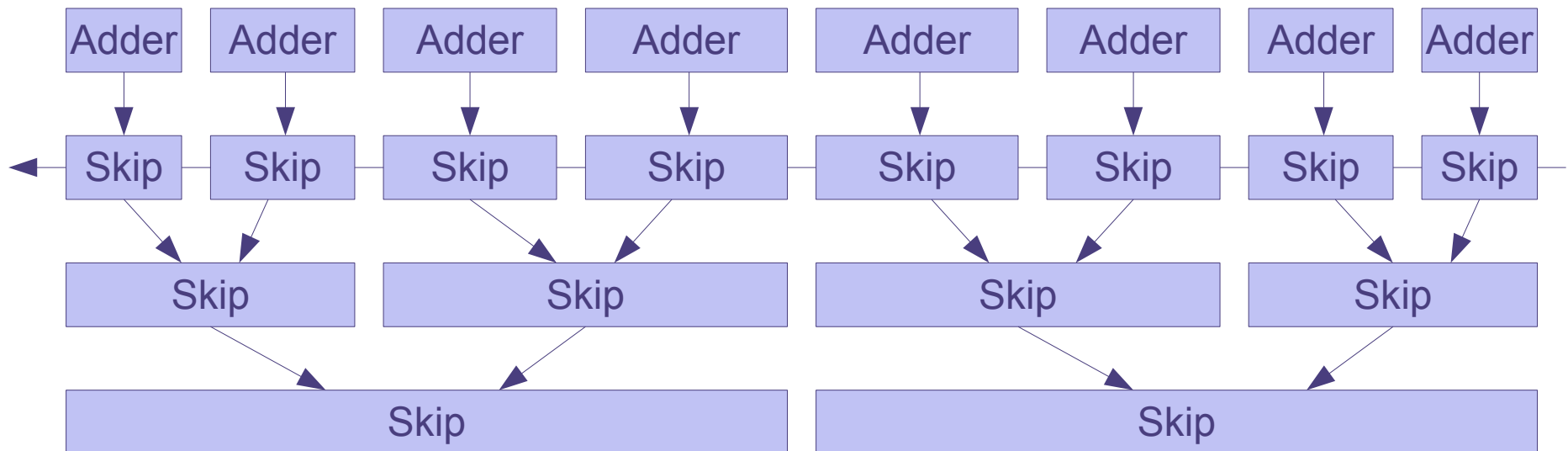
- 16-bit adder $\rightarrow t_{opt} = 8, T_{fixCSKA} \approx 5.5$

- e.g. 32-bit adder $\rightarrow t_{opt} \approx 12, T_{fixCSKA} \approx 9$

- e.g. 64-bit adder $\rightarrow t_{opt} = 16, T_{fixCSKA} \approx 13.5$

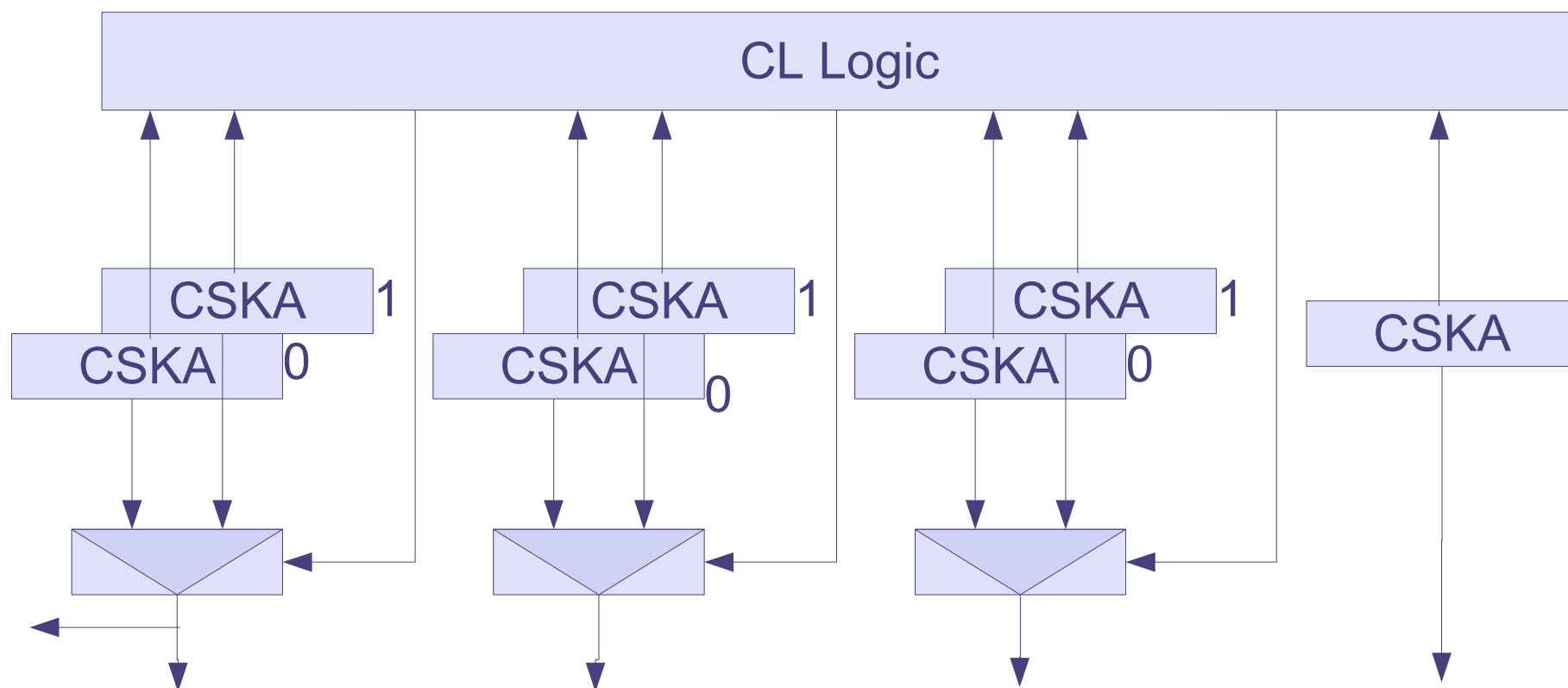
Wielopoziomowe sumatory CSKA

- Sprzęt pierwszego poziom przeskoku otrzymuje dane z sumatorów, drugi poziom z pierwszego, itd.
- Przeskok przeniesienia może odbywać się na znacznie dłuższych dystansach, co skraca opóźnienie dla najgorszego przypadku



Architektury hybrydowe

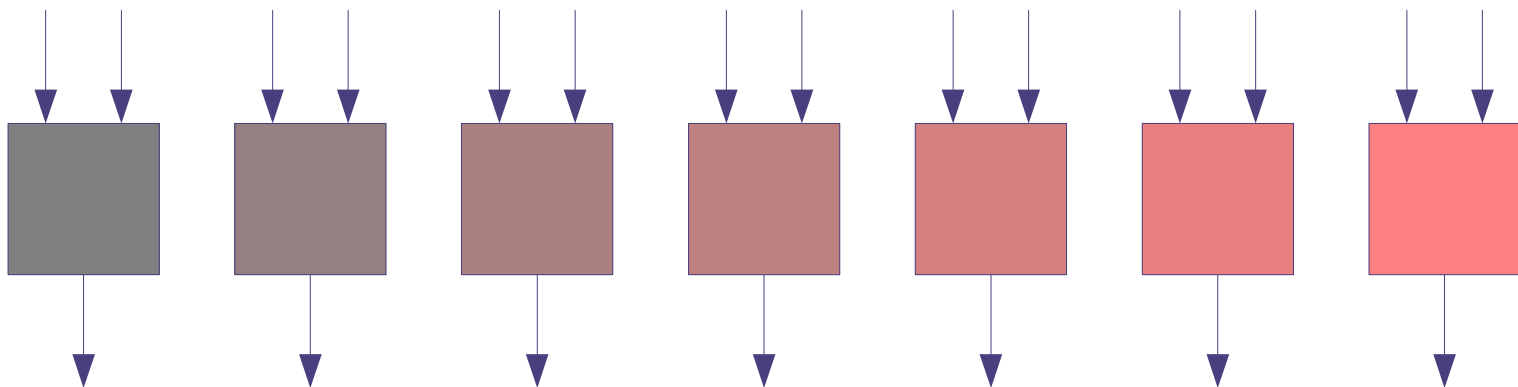
- Przykład: CSLA + CLA + CSKA



Dodawanie bez propagacji przeniesienia

- Dodawanie bez przeniesienia (*carry-free*) jest możliwe przy przejściu do redundantnych systemów liczbowych, z poszerzonym zestawem cyfr
- Operacje na wszystkich cyfrach są wykonywane całkowicie równoległe – szybkość: $\Theta(1)$

	1	2	3	4	5	6	
+	4	5	6	7	8	9	radix-10, zestaw cyfr [0,9]
	5	7	9	11	13	15	radix-10, zestaw cyfr [0,9]
							radix-10, zestaw cyfr [0,18]



Konwersja w systemach redundantnych

- Konwersja z notacji redundantnej do normalnej wymaga propagacji przeniesienia
- Szybkość: $\Theta(k)$

11	9	17	10	12	18		radix-10, zestaw cyfr [0,18]
11	9	17	10	12	18		18 = 10+8
11	9	17	10	13	8		13 = 10+3
11	9	17	11	3	8		11 = 10+1
11	9	18	1	3	8		18 = 10+8
11	10	8	1	3	8		10 = 10+0
12	0	8	1	3	8		12 = 10+2
1	2	0	8	1	3	8	radix-10, zestaw cyfr [0,9]

Reprezentacja liczb redundantnych

- Liczby redundantne mogą być wygodnie reprezentowane jako dwie liczby nieredundantne
- Dekmpozycja nie jest jednoznaczna, ale wartość całości jest jednoznaczna
- Konwersja liczby redundatnej wymaga po prostu dodania obu składników

	11	9	17	10	12	18	radix-10, zestaw cyfr [0,18]	
	9	9	9	9	9	9	radix-10, zestaw cyfr [0,9]	
+	2	0	8	1	3	9	radix-10, zestaw cyfr [0,9]	
	1	2	0	8	1	3	8	radix-10, zestaw cyfr [0,9]

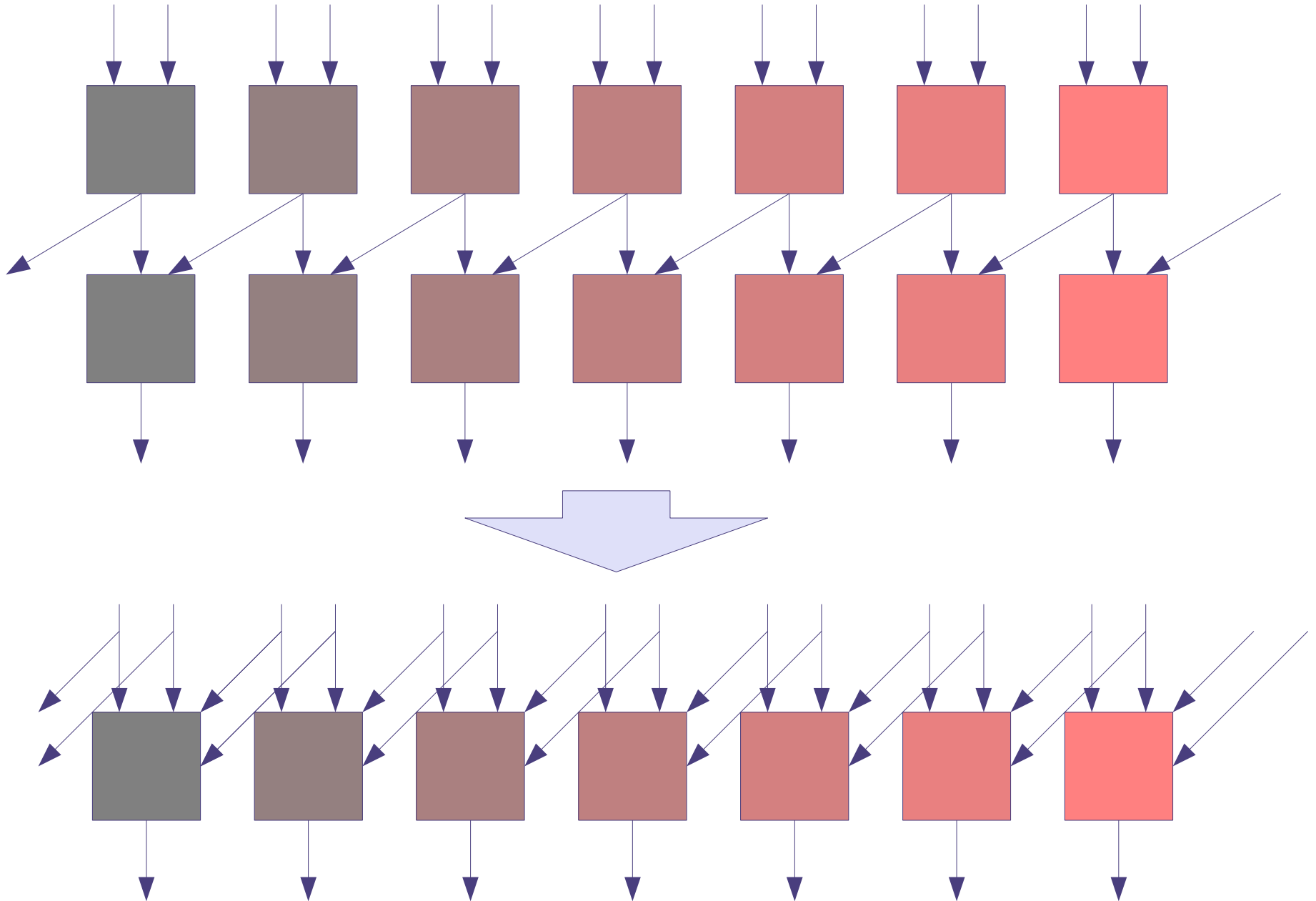
	0	1	2	1	1	1	radix-2, zestaw cyfr [0,2]
	0	0	1	0	0	1	radix-2, zestaw cyfr [0,1]
+	0	1	1	1	1	0	radix-2, zestaw cyfr [0,1]
	1	0	0	1	1	1	radix-2, zestaw cyfr [0,1]

Ograniczona propagacja przeniesienia

- W systemach redundantnych, oprerandy i wynik mogą mieć ten sam zestaw cyfr
- Mozliwa jest redukcja sum pośrednich za pomocą przeniesień, ograniczonych tylko do jednej pozycji w przód (*limited-carry*) – szybkość: $\Theta(1)$

$ \begin{array}{r} 11 9 17 10 12 18 \\ + 6 12 9 10 8 18 \\ \hline 17 21 26 20 20 36 \\ \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\ 7 11 16 0 10 16 \\ \swarrow \swarrow \swarrow \swarrow \swarrow \swarrow \\ 1 1 1 2 1 2 \\ \hline 1 8 12 18 1 12 16 \end{array} $	<p>radix-10, zestaw cyfr [0,18]</p> <p>radix-10, zestaw cyfr [0,18]</p> <hr style="border: 0.5px solid black;"/> <p>radix-10, zestaw cyfr [0,36]</p> <p>Cyfry pozycyjne [0,16]</p> <p>Cyfry przeniesień [0,2]</p> <hr style="border: 0.5px solid black;"/> <p>Wynik końcowy (suma) [0,18]</p>
---	---

Realizacja ograniczonej propagacji



Limited-carry w systemie dwójkowym

- Wielokrotne dodawanie liczb: $[0,2] + [0,1] \rightarrow [0,2]$

$$\begin{array}{r}
 \quad \mathbf{0 \ 1 \ 2 \ 1 \ 1 \ 1} \\
 + \quad \mathbf{0 \ 1 \ 1 \ 1 \ 0 \ 1} \\
 \hline
 \end{array}$$

radix-2, zestaw cyfr $[0,2]$

radix-2, zestaw cyfr $[0,1]$

radix-2, zestaw cyfr $[0,3]$

$$\begin{array}{r}
 \quad 0 \ 2 \ 3 \ 2 \ 1 \ 2 \\
 \quad \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
 \quad 0 \ 0 \ 1 \ 0 \ 1 \ 0 \\
 \quad \swarrow \swarrow \swarrow \swarrow \swarrow \swarrow \\
 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0
 \end{array}$$

Cyfry pozycyjne $[0,1]$

Cyfry przeniesień $[0,1]$

$$\begin{array}{r}
 \quad \mathbf{1 \ 1 \ 2 \ 0 \ 2 \ 0} \\
 + \quad \mathbf{0 \ 0 \ 1 \ 0 \ 1 \ 1} \\
 \hline
 \end{array}$$

radix-2, zestaw cyfr $[0,2]$

radix-2, zestaw cyfr $[0,1]$

radix-2, zestaw cyfr $[0,3]$

$$\begin{array}{r}
 \quad 1 \ 1 \ 3 \ 0 \ 3 \ 1 \\
 \quad \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
 \quad 1 \ 1 \ 1 \ 0 \ 1 \ 1 \\
 \quad \swarrow \swarrow \swarrow \swarrow \swarrow \swarrow \\
 0 \ 0 \ 1 \ 0 \ 1 \ 0
 \end{array}$$

Cyfry pozycyjne $[0,1]$

Cyfry przeniesień $[0,1]$

$$\begin{array}{r}
 \quad \mathbf{1 \ 2 \ 1 \ 1 \ 1 \ 1} \\
 \quad \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \\
 \hline
 \end{array}$$

radix-2, zestaw cyfr $[0,2]$

Konwersja do NKB

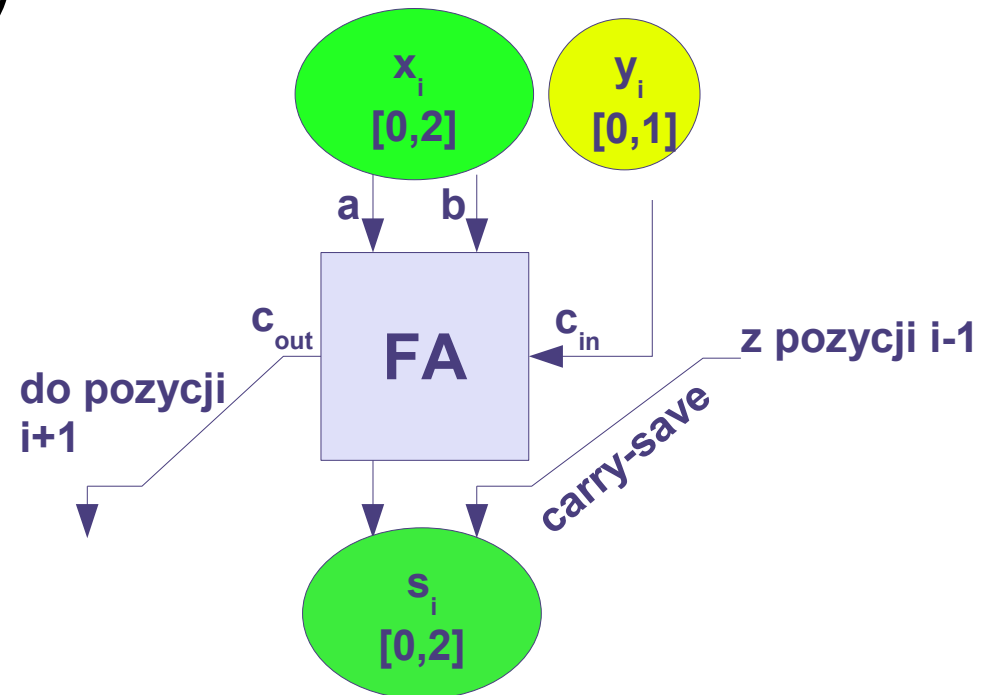
$$\begin{array}{r}
 \mathbf{1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1}
 \end{array}$$

Końcowy wynik dodawania (NKB)

Sumator z zachowaniem przeniesień

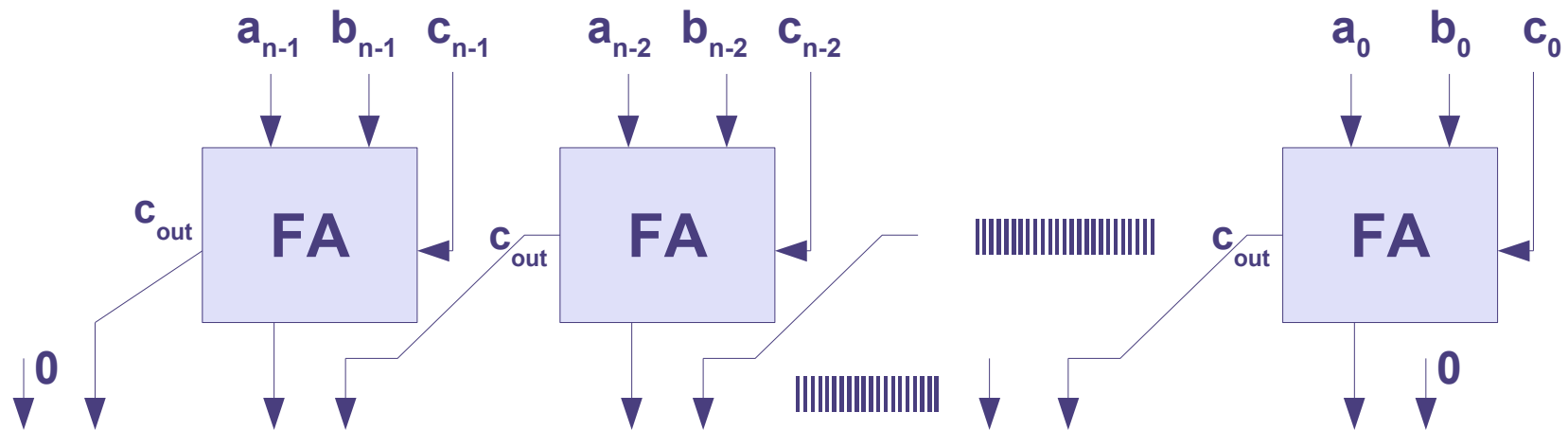
- Liczby redundantne $[0,2]$ przechowywane są w postaci dwóch liczb NKB: pozycyjnej i przeniesień
- Elementem wykonującym dekompozycję liczby $[0,3]$ $\rightarrow [0,1] + [0,1]$ jest pełny sumator (FA) w układzie CSA (Carry-Save Adder)

	0	1	2	1	1	1	$x [0,2]$
+	0	1	1	1	0	1	$y [0,1]$
	0	2	3	2	1	2	$[0,3]$
	↓	↓	↓	↓	↓	↓	
	0	0	1	0	1	0	$[0,1]$
	↙	↙	↙	↙	↙	↙	
0	1	1	1	0	1	0	$[0,1]$
	1	1	2	0	2	0	$s [0,2]$

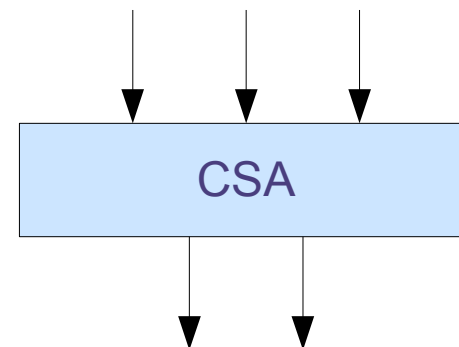


Sumator CSA

- Dodaje 3 liczby w NKB i daje w wyniku 2 liczby NKB
- Szybkość: $\Theta(1)$

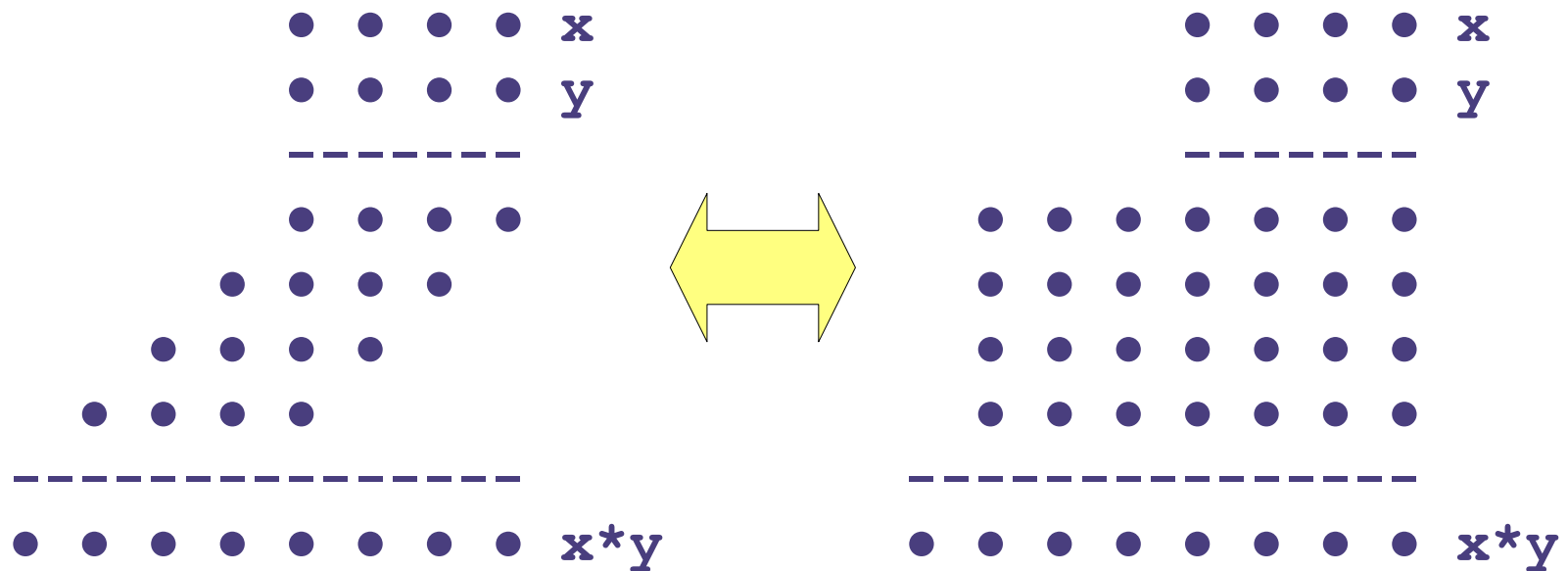


- Układ redukujący 3/2
(*bit-counter 3/2*)



Sumatory wieloargumentowe

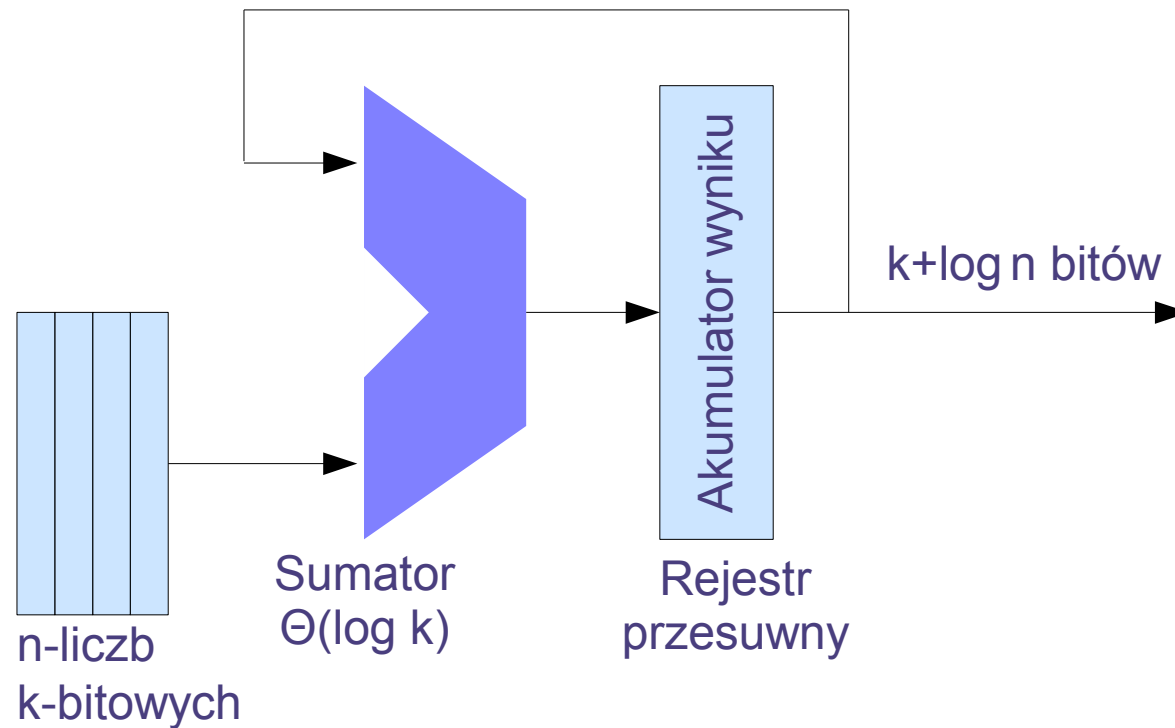
- Zastosowania: mnożenie, operacje na wektorach i macierzach



Dodawanie n -liczb k -bitowych daje wynik $k + \log_2 n$ bitowy

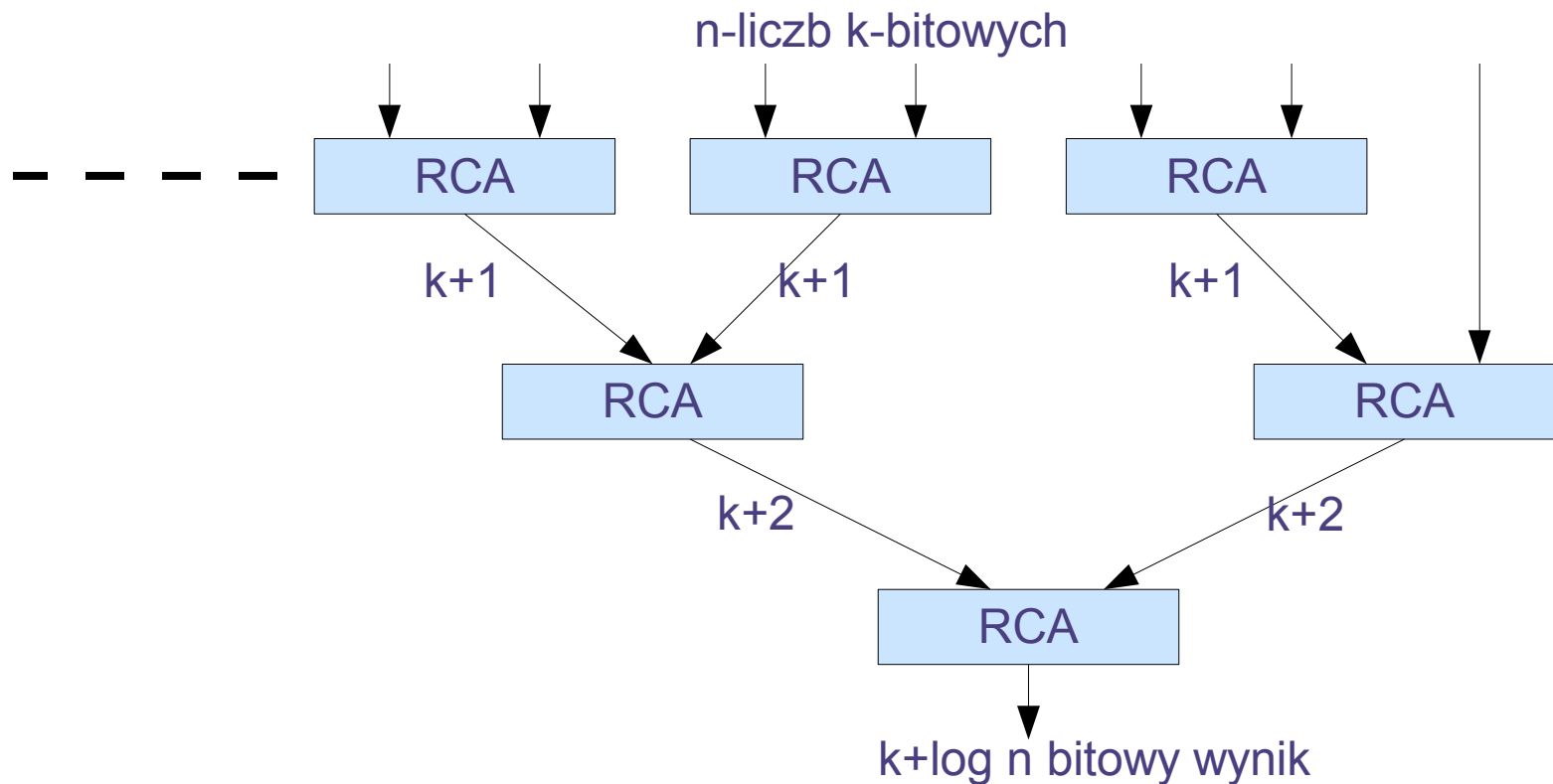
Implementacja szeregową

- Szybkość działania: $\Theta(n \log k)$



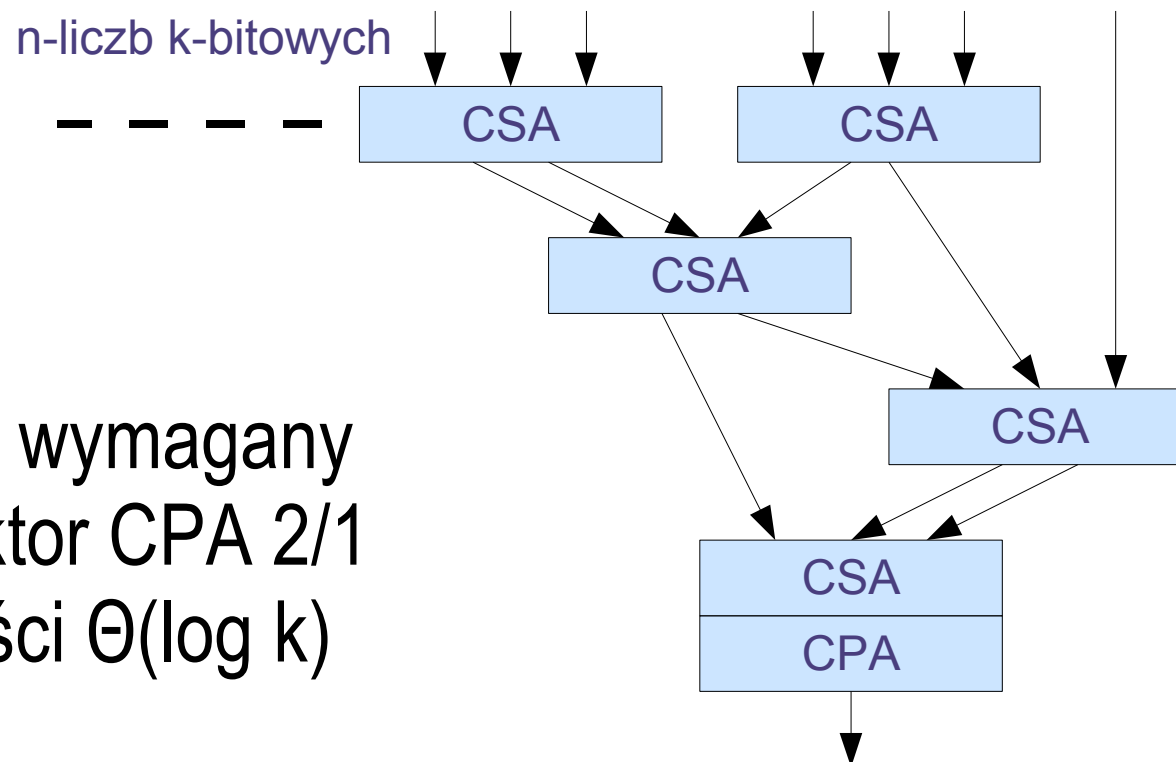
Drzewo sumatorów typu CPA (2/1)

- Każdy sumator typu CPA redukuje liczbę operandów
- Dla n -liczb wymagane jest $n-1$ sumatorów (dużo!)
- Szybkość: $\Theta(k + \log n)$ dla sumatorów RCA



Drzewo sumatorów typu CSA (3/2)

- Sumatory CSA redukują liczbę argumentów do dwóch
- Szybkość działania sumatora CSA wynosi $\Theta(1)$
- Szybkość całego drzewa: $\Theta(\log n + \log k)$



- Na końcu wymagany jest reduktor CPA 2/1 o szybkości $\Theta(\log k)$