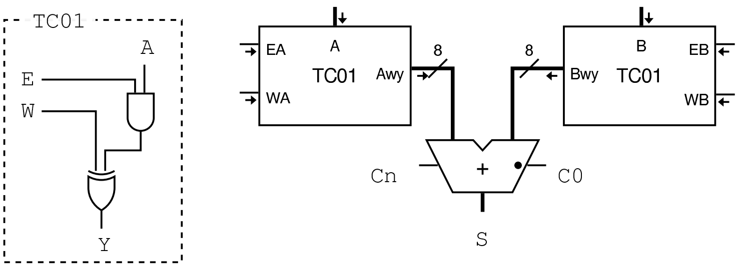
ELiAK 2			
Kierunek *Informatyka*	**Specjalność** *Informatyka*	**Rok studiów** *2*	**Symbol grupy lab.** *L04*
Temat Laboratorium *Sumatory*			**Numer lab.** *6*
Skład grupy ćwiczeniowej *Katarzyna Michoń, Elżbieta Wolanin, Dariusz Ostrowski, Wojciech Muła*			
Uwagi			**Ocena**

Ćwiczenie 6a. Podać pełną tablicę funkcji dla ALU.

ALU



Tablica prawdy dla układu „TC01”

E	W	Y
0	0	$0\ldots0$
1	0	A
0	1	$1\ldots1$
1	1	$\bar{A}$

Funkcje *ALU*

$Y_A(E_A, W_A)$	$Y_B(E_B, W_B)$	C_{-1}	S	C	C_{-1}	S	C
$0\ldots0$	$0\ldots0$	0	$0\ldots0 = 0_{U2}$	0	1	1	0
$0\ldots0$	$1\ldots1$	0	$1\ldots1 = -1_{U2}$	0	1	0	1
$1\ldots1$	$0\ldots0$	0	-1_{U2}	0	1	0	1
$1\ldots1$	$1\ldots1$	0	$1\ldots10 = -2_{U2}$	1	1	-1	1
A	$0\ldots0$	0	A		1	$A+1$ (INC A)	
$\bar{A}$	$0\ldots0$	0	$\bar{A}$ (NOT A)		1	U2: $\bar{A}+1 = -A$ (NEG A)	
A	$1\ldots1$	0	U2: $A-1$ (DEC A)		1	U2: $A-1+1 = A$	1
$\bar{A}$	$1\ldots1$	0	U2: $\bar{A}-1$		1	U2: $\bar{A}-1+1 = \bar{A}$	1
$0\ldots0$	B	0	B		1	$B+1$ (INC B)	
$0\ldots0$	$\bar{B}$	0	$\bar{B}$ (NOT B)		1	U2: $\bar{B}+1 = -B$ (NEG B)	
$1\ldots1$	B	0	U2: $B-1$ (DEC B)		1	U2: $B-1+1 = B$	1
$1\ldots1$	$\bar{B}$	0	U2: $\bar{B}-1$		1	U2: $\bar{B}-1+1 = \bar{B}$	1
A	B	0	$A+B$		1	$A+B+1$	
$\bar{A}$	B	0	$\bar{A}+B$		1	U2: $\bar{A}+1+B = B-A$	
A	$\bar{B}$	0	$A+\bar{B}$		1	U2: $A+\bar{B}+1 = A-B$	
$\bar{A}$	$\bar{B}$	0	$\bar{A}+\bar{B}$		1	U2: $\bar{A}+\bar{B}+1$	

Ćwiczenie 6b. Zaprojektować 4-bitowy sumator *carry-look-ahead*

Funkcje opisujące sumator są następujące:

- i-ty bit wyniku: $s_i = p_i \oplus c_{i-1}$
- Przeniesienie z i-tej pozycji: $c_i = g_i + p_i c_{i-1}$

gdzie $g_i = a_i b_i$ (grupa generacyjna), $p_i = a_i \oplus b_i$ (grupa propagacyjna).

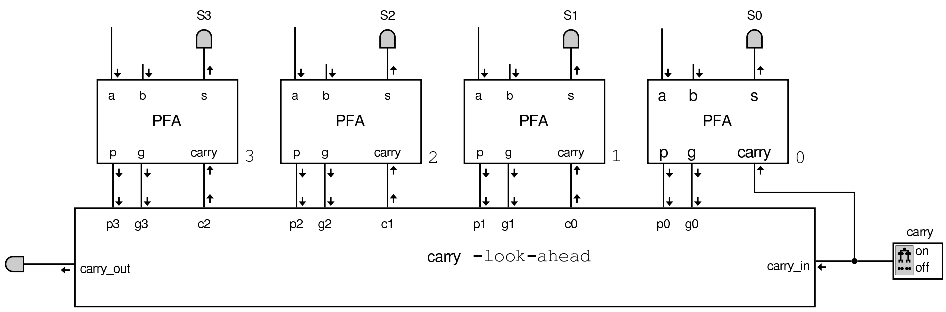
Dla wszystkich bitów sumy $s_0, \ldots, s_3$ zostanie użyta ww funkcja, natomiast funkcje opisujące bity przeniesienia zostaną rozpisane.

- $c_0 = g_0 + p_0 c_{-1}$
- $c_1 = g_1 + p_1 c_0 = g_1 + p_1(g_0 + p_0 c_{-1}) = g_1 + p_1 g_0 + p_1 p_0 c_{-1}$
- $c_2 = g_2 + p_2 c_1 = g_2 + p_2(g_1 + p_1(g_0 + p_0 c_{-1})) = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_{-1}$
- $c_3 = g_3 + p_3 c_2 = g_3 + p_3(g_2 + p_2(g_1 + p_1(g_0 + p_0 c_{-1}))) = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_{-1}$

Układ został zbudowany z dwóch bloków funkcjonalnych:

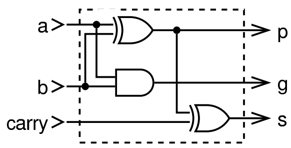
1. **PFA** (ang. *Partial Full Adder*), w którym obliczane są wartości sygnałów p_i, g_i, oraz s_i;
2. generator przeniesień równoległych (ang. *carry-look-ahead*), który wyznacza wartości przeniesień $c_0, \ldots, c_3$.

Schemat układu

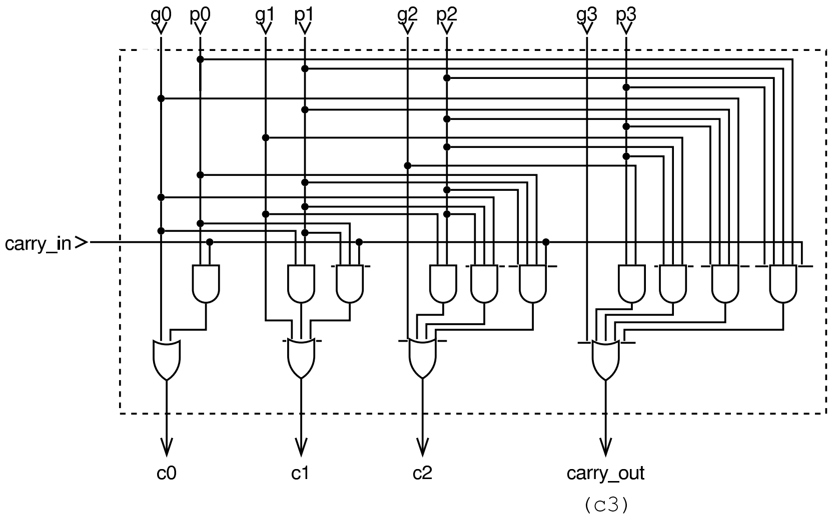


Schemat bloku *PFA*

```
g = ab
p = a xor b
s = p xor carry
```



Schemat generatora przeniesień równoległych

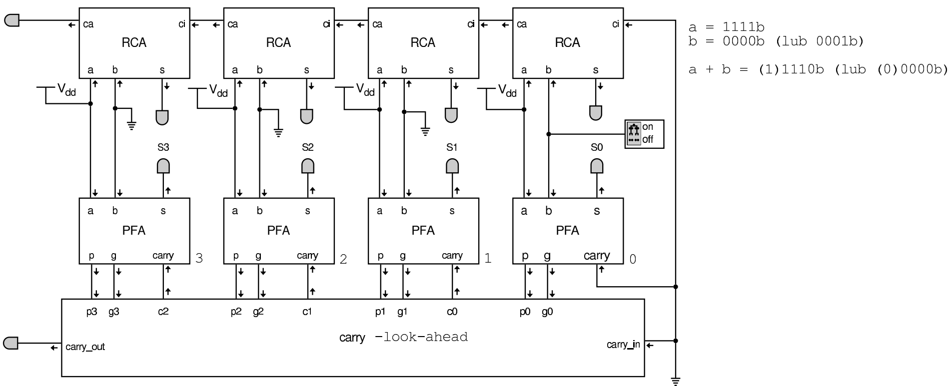


Wnioski

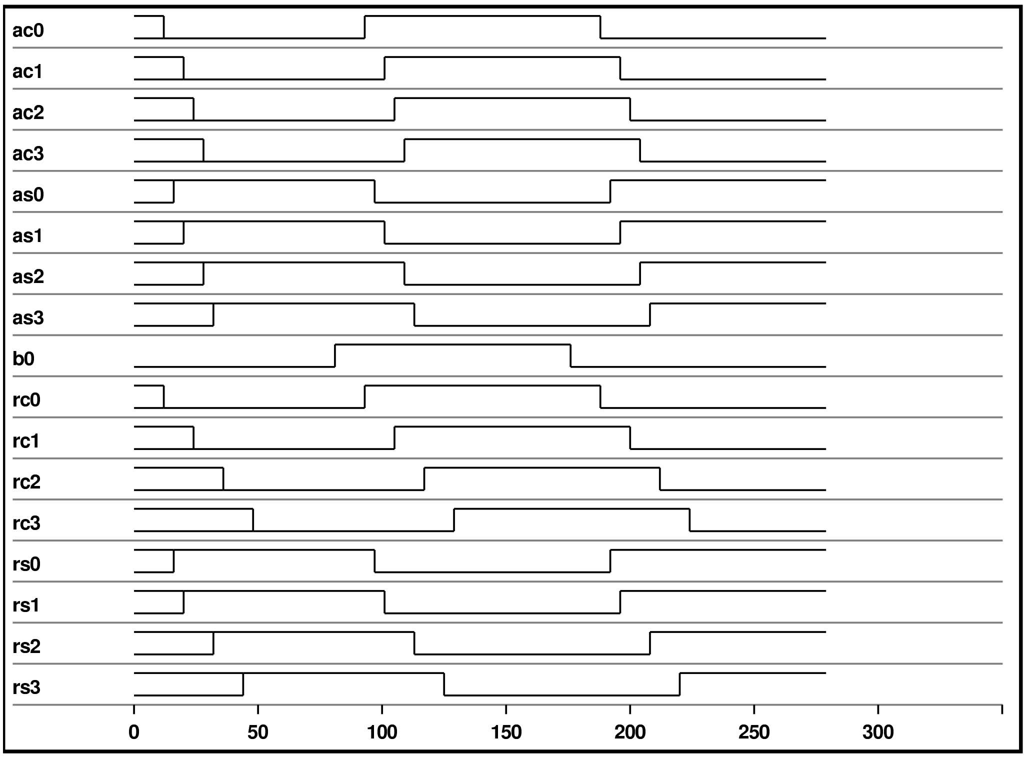
- Za pomocą ALU można zrealizować operacje dodawania, odejmowania, zmniejszania/zwiększania o jeden, negować bitowo słowa, itp. Można również generować stałe: 0, 1, -1, -2. Niektóre operacje można zrealizować za pomocą różnych kombinacji sygnałów sterujących.

- W sumatorze *carry-look-ahead* czas propagacji przeniesień jest (praktycznie) stały, niezależny od pozycji w słowie bitowym. Okupione jest to jednak skomplikowaniem układu generującego przeniesienia.

  Poniżej porównanie działania szeregowego sumatora (ang. *ripple-carry adder*) oraz badanego sumatora. Oba sumatory (równocześnie) dodawały dwie liczby: 1111_b i 0000_b albo 1111_b i 0001_b (najmłodszy bit słowa b był sterowany przełącznikiem $b0$). Na przebiegach sygnały $ac0, \ldots, ac3$ (przeniesienie) i $as0, \ldots, as3$ (suma) odnoszą się do sumatora z przeniesieniami równoległymi, natomiast $rc0, \ldots, rc3$ i $rs0, \ldots, rs3$ to sygnały z sumatora z przeniesieniami szeregowymi.



Standardowo, w programie *TkGate*, czas propagacji sygnałów w bramkach zależy od liczby wejść (co jest bliższe rzeczywistości): dla bramki 2-wejściowej wynosi 10 jednostek czasu symulatora, dla 3-wejściowej 12, dla 4-wejściowej 14, itd. $(10 + (n-2) \cdot 2)$.



Widać wyraźnie, że od chwili zmiany *b*0 czas potrzebny na zmianę wszystkich sygnałów *axx* jest mniejszy niż sygnałów *bxx*. Prawidłowe wartości, jako ostatnie, przyjmują bity przeniesienia *ac*3 i *rc*3. Przy czym dla *ac*3 dzieje się to po upływie 30-32 jednostek czasu, zaś dla *rc*3 po 45-48 jednostkach – 30% procent wolniej.