Lista 1

zad	1	2	3	4	5	6	7
pkt.	1	1	1	1	1	1	1

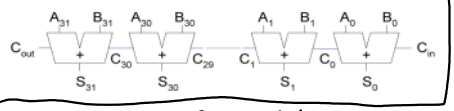
Zadanie 1. Rozważmy n-bitowy sumator z przeniesieniem szeregowym (ang. *ripple-carry adder*, RCA). Sumuje on dwie liczby n-bitowe bez znaku, dając w wyniku również liczbę n-bitową.

a) • Zakładając, że każda bramka działa w czasie jednostkowym, jaki jest czas działania tego sumatora (tzn. ile jednostek czasu potrzeba by na wyjściach sumatora pojawił się poprawny wynik dodawania)?

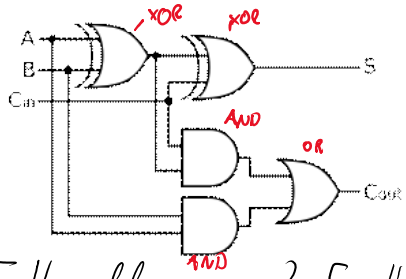
b) • W jaki sposób wykryć, że wynik sumowania nie mieści się w n bitach (przepełnienie)?

a)

- Chain 1-bit adders together
- Carry ripples through entire chain
- Disadvantage: slow



łączymy n full-odderów



Full odder ma 3 „fazy" bramek,
z założenia każda bramka = jednostka czasu

Musimy przejść po kolei każdy odder = łącznie 3n

b) Przepełnienie → cout = 1

Zadanie 2. Pokaż, że n-bitowy sumator RCA poprawnie wykonuje operację sumowania, jeśli argumenty interpretujemy jako liczby ze znakiem w kodzie uzupełnień do dwóch.

Overflow przy dwoch dodatnich = znak; przy dwoch ujemnych -> znak na 0 i cout na 1

Liczby U2 poprawnie się sumują pisemnie
RCA działa tak samo jak dodawanie pisemne
xor bitów i przeniesienie wykorzystujemy jako wynik
alternatywę koniunkcji par bitów jako carry-over

Zadanie 3. Zaprojektuj układ kombinacyjny (w postaci tabelki oraz grafu bramek), który wykrywa przepełnienie podczas dodawania liczb n-bitowych w kodzie uzupełnień do dwóch, za pomocą sumatora RCA. Układ powinien wyprowadzać na wyjście wartość 01, jeśli nastąpiło przeniesienie dodatnie (suma jest liczbą dodatnią niemieszczącą się w n bitach), wartość 10 jeśli nastąpiło przeniesienie ujemne (suma jest liczbą ujemną niemieszczącą się w n bitach) oraz wartość 00 w przeciwnym przypadku.

a_n, b_n - najstarsze bity dodawanych liczb (1 - lb. ujemna, 0 dodatnia)

c_{n-1} - wyjście ostatniego przeniesienia

Chcemy otrzymać:

00 jeśli: a,b różnych znaków lub a,b,c tego samego

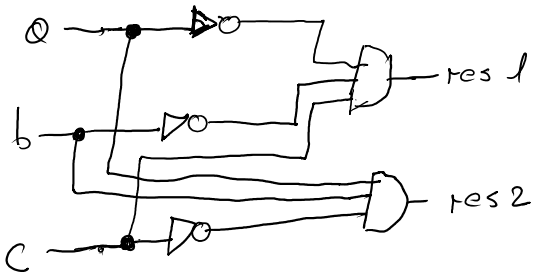
01: a,b ujemne, c dodatnie

10: a,b dod, c ujemne

				result	
a	b	c		c.1.	c.2.
0	0	0		0	0
0	0	1		0	1
0	1	0		0	0
0	1	1		0	0
1	0	0		0	0
1	0	1		0	0
1	1	0		1	0
1	1	1		0	0

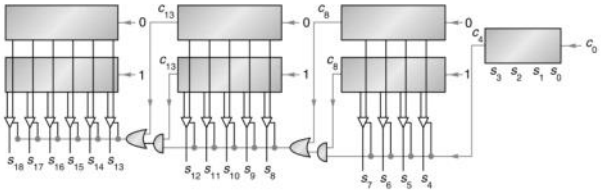
c.1: $\neg a \wedge \neg b \wedge c$

c.2: $a \wedge b \wedge \neg c$



Zadanie 4. Na rysunku poniżej przedstawiono 19-bitowy sumator z wyborem przeniesień (ang. *carry-select adder*). Sumator ten składa się z siedmiu sumatorów RCA (szare bloki), multiplekserów (trójkąty) oraz bramek. Każdy sumator RCA przyjmuje, oprócz bitów do zsumowania, pewne przeniesienie wejściowe. Produkuje natomiast, oprócz bitów sumy, przeniesienie wyjściowe. Każdy z trzech par sumatorów RCA liczy swój fragment sumy, przy założeniu, że przeniesienie wejściowe jest równe odpowiednio 0 lub 1. Następnie, po ustaleniu faktycznej wartości tego przeniesienia, multipleksery wybierają odpowiednie bity do wyprowadzenia na

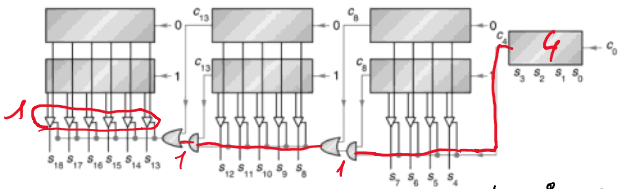
wyjście jako bity sumy. Produkowane jest też przeniesienie dla następnej pary sumatorów RCA.



Przeanalizuj działanie tego układu i upewnij się, że dobrze je rozumiesz. Wylicz jego czas działania. Następnie załóż, że mamy zaprojektować n-bitowy sumator według tego schematu, przy czym wszystkie sumatory RCA mają mieć tą samą liczbę wyjść k. Jaka powinna być wartość k, by czas działania zaprojektowanego układu był możliwie najmniejszy?

Wskazówka: "Appendix J", str. 43.

a)



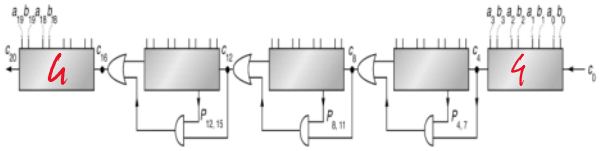
4+1+1+1 = 7 - tylko no to czekamy, reszta in meantime

b) Czas zależy od inicjalnego k, liczby bramek $(\frac{n}{k}-2)$ i 1 na obliczenie ostatnich multiplekserów

$T(k) = k + \frac{n}{k} - 2 + 1 = k + \frac{n}{k} - 1$

min: $T'(k) = 1 - \frac{n}{k^2} \Rightarrow k^2 = n \Rightarrow k = \sqrt{n}$

Zadanie 5. Powtórz zadanie 4. dla sumatora z przeskokiem przeniesień (ang. carry-skip adder) przedstawionego poniżej.



3

Szare bloki oznaczają sumatory RCA z dodatkowym układem liczącym wartości propagacji przeniesienia przez dany blok $P_{i,j}$, gdzie $P_{i,j} = p_i p_{i+1} p_{i+2} \cdots p_j$, a każda z wartości p_k jest propagacją przeniesienia przez pozycję k ($p_k = a_k + b_k$). Np. $P_{4,7}$ oznacza iloczyn logiczny $p_4 p_5 p_6 p_7$, a $p_4 = a_4 + b_4$.

Wskazówka: "Appendix J", str. 41.

a) 4 start, 4 koniec, 3 z bramek $\Rightarrow \wedge\wedge$

b) Optymalizacja analogiczna do 4a

$T(k) = k + \frac{n}{k} - 2 + k = 2k + 2\frac{n}{k} - 4$

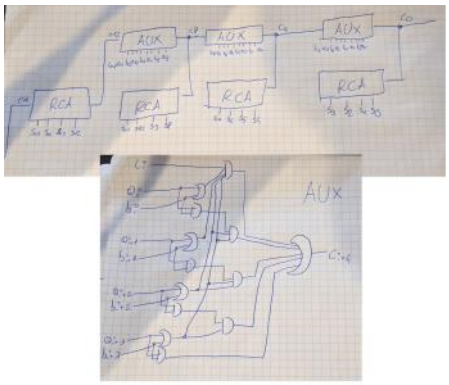
min: $T'(k) = 2 - \frac{n}{k^2}$

$2n = k^2 \quad k = \sqrt{n}$

Zadanie 6. Opracuj sumator złożony z bloków (podobnie jak w poprzednim zadaniu), w którym każdy blok jest sumatorem RCA z dodatkowym układem obliczającym przeniesienie z tego bloku. Ten dodatkowy układ może korzystać jedynie z przeniesienia wejściowego danego bloku oraz bitów argumentów do zsumowania w danym bloku. Ideę tego układu oprzyj na liczeniu propagacji

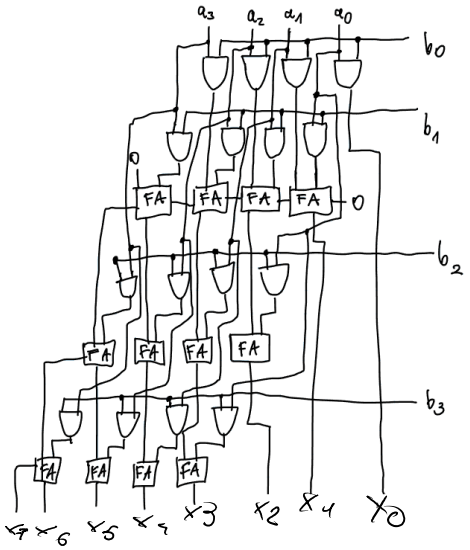
Zadanie 6. Opracuj sumator złożony z bloków (podobnie jak w poprzednim zadaniu), w którym każdy blok jest sumatorem RCA z dodatkowym układem obliczającym przeniesienie z tego bloku. Ten dodatkowy układ może korzystać jedynie z przeniesienia wejściowego danego bloku oraz bitów argumentów do zsumowania w danym bloku. Ideę tego układu oprzyj na liczeniu propagacji przeniesienia i generowania przeniesienia przez dany blok. Będzie to tzw. blokowy sumator z przeniesieniem równoległym (ang. *carry-lookahead adder*, CLA). Narysuj swój n-bitowy sumator dla n=16 i rozmiaru bloku k=4.

Wskazówka: "Appendix J", str. 37.



Zadanie 7. Bazując na algorytmie mnożenia pisemnego, opracuj układ mnożący liczby 4-bitowe. Jako podstawowe elementy układu możesz użyć dowolnych bramek i sumatorów.

Wskazówka: np. "Appendix J", str. 50



10·10·11=110

1011
·1010
0000
1011
0000
+1011
1101110