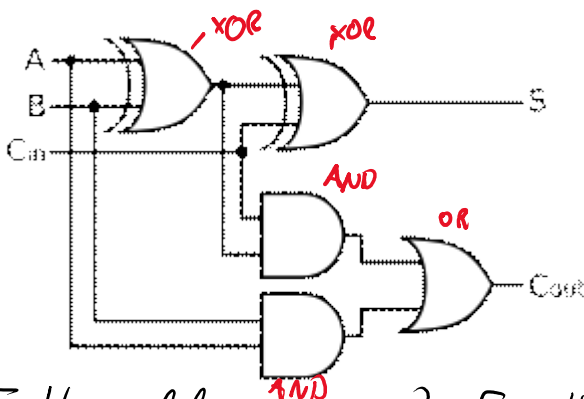
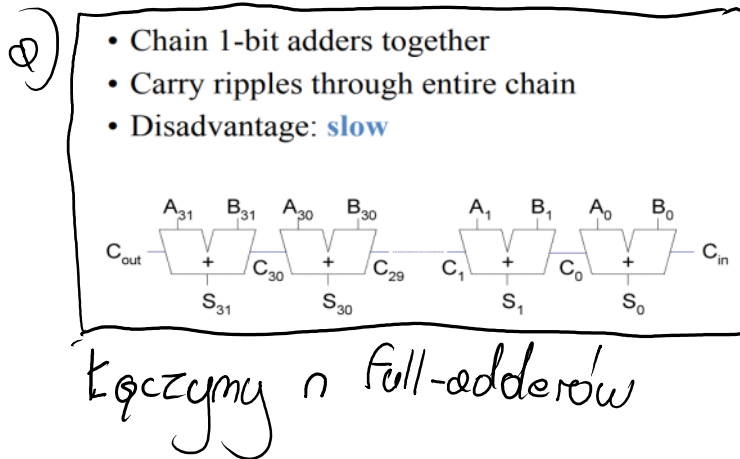


Lista 1

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)?



Full adder ma 3 „fazy” bramek,
z założenia każda bramka = jednostka czasu

Musimy przejść po kolei każdy adder = łącznie **$3n$**

b) Przepełnienie \rightarrow **$c_{out} = 1$**

b) Przepetnienie \rightarrow 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.

Liczby U2 poprawnie się sumują pisemnie
RCA dzięki temu samo jak dodawanie pisemne
xor bitów i przeniesienie wpisujemy jako wynik
dysjunkcji koniunkcji pos 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 nie mieszczącą się w n bitach), wartość 10 jeśli nastąpiło przeniesienie ujemne (suma jest liczbą ujemną nie mieszczącą się w n bitach) oraz wartość 00 w przeciwnym przypadku.

a_n, b_n - najstarsze bity dodawanych liczb (1-1b. ujemne, 0 dodatnie)
 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 dodatnie, c ujemne

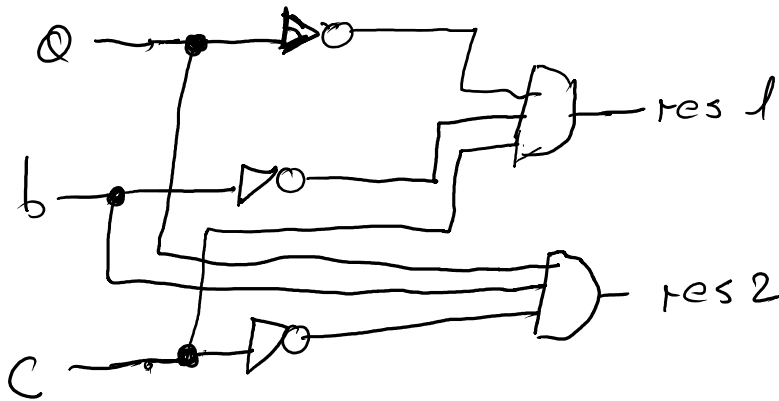
			result	
a	b	c	c.1.	c.2.
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	0

c.1: $\neg 0 \vee b \wedge c$

c.2: $0 \wedge b \vee 1 \wedge c$

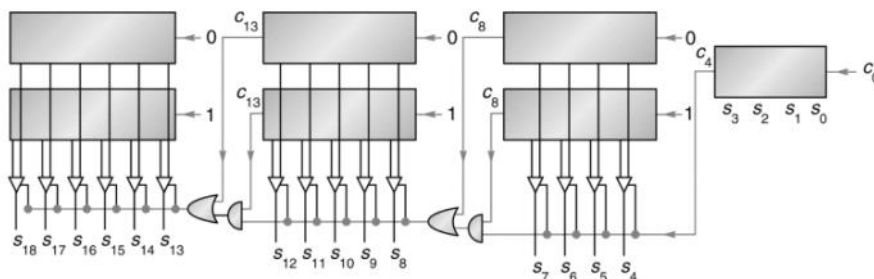
0	0	0		0	0
0	0	1		1	0
0	1	0		0	0
0	1	1		0	0
1	0	0		0	0
1	0	1		0	0
1	1	0		0	1
1	1	1		0	0

C, 2% 0 1 6 1 + C



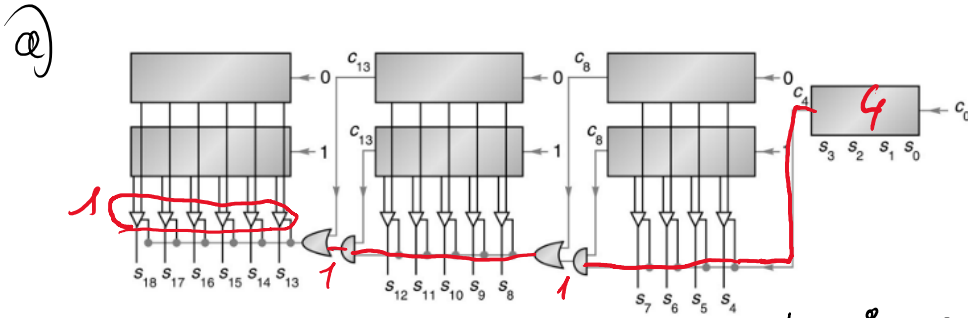
Zadanie 4. Na rysunku poniżej przedstawiono 19-bitowy sumator z wyborem przeniesień (ang. carry-select adder). Sumator ten składa się z siedmiu sumatorów RCA (szare bloki), multiplexerów (trójkąty) oraz bramek. Każdy sumator RCA przyjmuje, oprócz bitów do zsumowania, pewne przeniesienie wejściowe. Produkuje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

wyjscie 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.



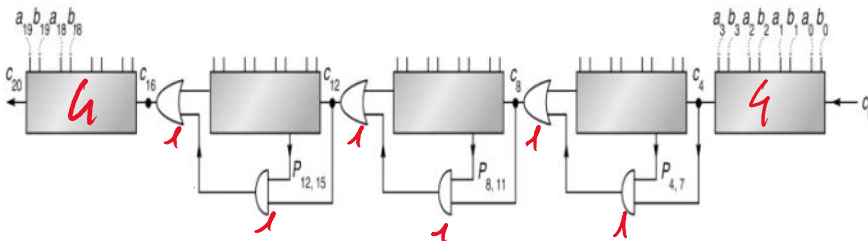
$4+1+1+1=7$ - tylko no to czekamy, reszta in meantime

b) Czas zależy od innego k, liczby bloków $(\frac{n}{k}-2)$ i od obciążenia ostatnich multiplexerów

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

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

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



Szare bloki oznaczają sumatory RCA z dodatkowym układem liczącym wartości propagacji przeniesienia przez dany blok P_{ij} , gdzie $P_{ij} = p_i p_{i+1} p_{i+2} \dots p_j$, 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*2 z blokiem $\Rightarrow 14$

b) Optymalizacja analityczna do 4:

$$T(k) = k + (\frac{n}{k}-2) \cdot 2 + 1 = 2\frac{n}{k} + 2k - 4$$

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

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

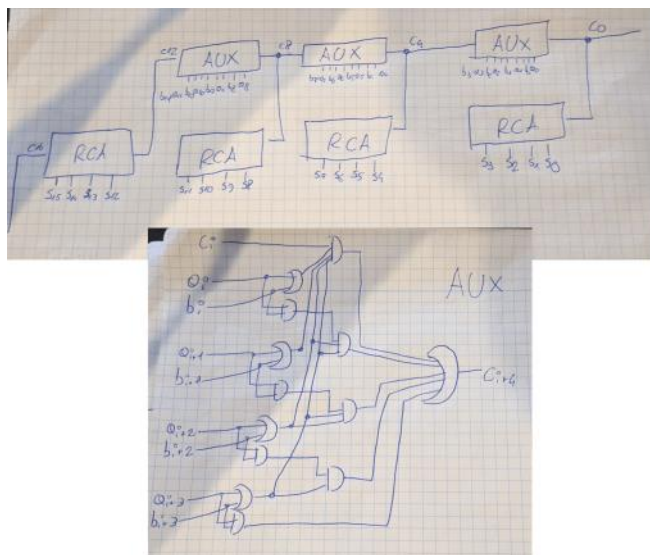
$$T(k) = k^2 \sqrt{n}$$

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

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

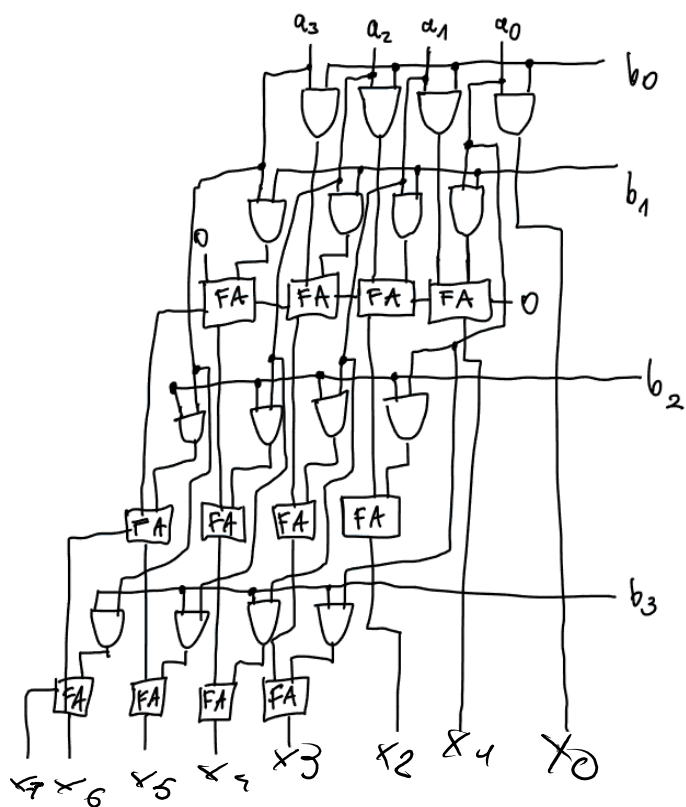
Zadanie 6. Opracuj sumator złożony z bloków (podobnie jak w poprzednim zadaniu), w którym każdy blok jest sumatorem RCA z dodatkowym układem obliczającym przeniesienie z tego bloku. Ten dodatkowy układ może korzystać jedynie z przeniesienia wejściowego danego bloku oraz bitów argumentów do zsumowania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



$$10: 10 - 11 = 110$$

$$\begin{array}{r}
 2: 1011 \\
 - 1010 \\
 \hline
 0000 \\
 1011 \\
 0000 \\
 + 1011 \\
 \hline
 1101110
 \end{array}$$