Dodawanie liczb dwójkowych. Sumator.

Algorytmy dodawania liczb dziesiętnych i dwójkowych są podobne:

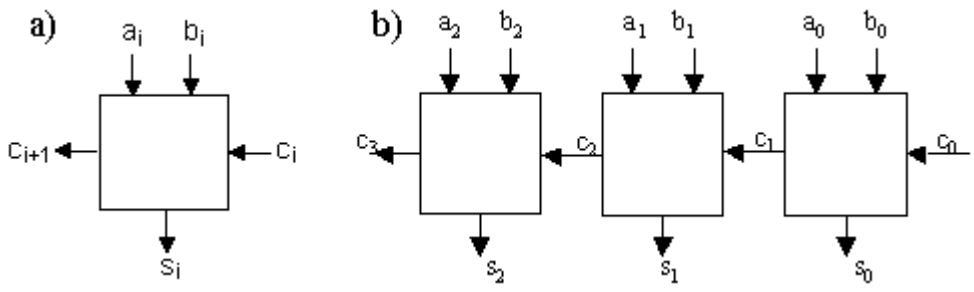
	binarnie	dziesiętnie
przeniesienia	1 1 1 0 1	1
składnik	0 1 1 0 1	1 3
składnik	1 1 1 0 1	2 9
suma	1 0 1 0 1 0	4 2

Dodawanie przebiega w tylu krokach, ile cyfr mają dodawane liczby. Długość liczby krótszej możemy zrównać z długością liczby dłuższej, dopisując przed nią odpowiednią ilość zer. W pierwszym kroku dodajemy do siebie dwie cyfry występujące na najmniej znaczącej pozycji liczb. W każdym następnym kroku dodajemy trzy cyfry: dwie cyfry występujące na danej pozycji sumowanych liczb oraz przeniesienie z poprzedniej, niższej pozycji; tzn. cyfrę 1, jeśli przeniesienie wystąpiło, albo cyfrę 0, jeśli przeniesienie nie wystąpiło. Także pierwszy krok możemy traktować jako dodanie trzech cyfr, z tym, że tu cyfra „przeniesienia z niższej pozycji” zawsze ma wartość 0. W każdym kroku otrzymujemy dwie nowe cyfry: „wynik”, który zostanie zapisany na danej pozycji oraz przeniesienie, które zostanie dodane do sumy cyfr znajdujących się na kolejnej, bardziej znaczącej pozycji albo – gdy długość wyniku przekracza długość dodawanych cyfr – zostanie zapisane na wyższej pozycji.

Układ sumujący dwie liczby mógłby więc mieć postać łańcucha złożonego z wielu identycznych sumatorów elementarnych. W każdym sumatorze elementarnym byłyby sumowane trzy cyfry: dwie cyfry występujące się na tej samej pozycji liczb dodawanych oraz cyfra przeniesienia z niższej pozycji. Każdy sumator elementarny powinien posiadać trzy wejścia do podania na nie stanów reprezentujących trzy sumowane cyfry. Sumatorów elementarnych powinno być co najmniej tyle, ile cyfr zawierają sumowane liczby. Na jedno z wejść (konkretnie na wejście przeznaczone do podania na nie cyfry przeniesienia) w sumatorze elementarnym dodającym cyfry występujące na najmniej znaczącej pozycji powinna być stale podawana cyfra 0.

Oczywiście sumator elementarny dodający cyfry występujące na najmniej znaczącej pozycji może mieć inną (prostszą) budowę, niż pozostałe sumatory elementarne, gdyż może posiadać tylko dwa wejścia.

Na rys. 1 został pokazany symbol sumatora elementarnego oraz łańcuch złożony z trzech sumatorów elementarnych, mogący służyć do dodawania dwu trzybitowych liczb binarnych. Cyfra c_0 powinna mieć stale wartość 0.



Rys. 1. Symbol sumatora elementarnego (rys. a) oraz sumator dodający dwie liczby trzybitowe (rys. b); a_i, b_i – cyfry sumowanych liczb, c_i, c_{i+1} – przeniesienia, s_i – cyfry wyniku.

Sumator elementarny.

Reguły dodawania trzech cyfr w sumatorze elementarnym binarnym najłatwiej jest przedstawić w postaci tabelek podających cyfry wyniku (suma bez przeniesienia) i przeniesienia dla wszystkich wartości argumentów. Na rysunku 2 został pokazany szczególny rodzaj takich tabelek, tzw. tablice Karnaugha. Kolejne numery kolumn oraz wierszy przedstawione są w nich za pomocą kodu Gray'a. Zerom i jedynkom z kodu Gray'a odpowiadają wartości zmiennych a_i, b_i oraz c_i, Mając zapisaną w tablicy Karnaugha regułę działania układu (czyli funkcję, którą ma realizować układ logiczny), łatwo jest napisać postać zminimalizowaną wzoru logicznego (boolowskiego), jaki powinien realizować układ. Wystarczy wypisać sumy iloczynów kombinacji zmiennych i ich zaprzeczeń „generujących” pojedyncze izolowane jedynki oraz iloczynów pozbawionych zmiennych, dla których jedynki sąsiadujące ze sobą w tablicy są generowane dla obu (czyli dla 0 i 1) wartości tych zmiennych.

a)

c_i \ a_ib_i	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

cyfry wyniku s_i

b)

c_i \ a_ib_i	00	01	11	10
0	0	0	1	0
1	0	1	1	1

wartości przeniesienia c_{i+1}

Rys. 2. Sumator jednobitowy: tablice Karnaugha przedstawiające wartości wyniku s_i (rys. a) i przeniesienia c_{i+1} (rys. b) dla różnych wartości a_i, b_i i c_i.

Funkcje logiczne realizowane przez sumator elementarny wypisane na podstawie tablic Karnaugha z rys. 2 mają postacie:

$$s_i = \bar{a}_i\bar{b}_i c_i + a_i b_i c_i + \bar{a}_i b_i \bar{c}_i + a_i \bar{b}_i \bar{c}_i$$

$$c_{i+1} = a_i b_i + a_i c_i + b_i c_i$$

Kreski nad symbolami wartości oznaczają negacje tych wartości.

Wzór na wartość s_i da się napisać w postaci jeszcze krótszej, niż otrzymana na podstawie tablicy Karnaugha postać zminimalizowana, zawierająca sumy iloczynów. Stosując zasady algebry Boola napiszemy:

$$s_i = c_i(a_i b_i + \bar{a}_i\bar{b}_i) + \bar{c}_i(\bar{a}_i b_i + a_i\bar{b}_i)$$

Wyrażenie w pierwszym nawiasie jest negacją wyrażenia w drugim nawiasie:

$$a_i b_i + \bar{a}_i\bar{b}_i = \overline{\bar{a}_i b_i + a_i\bar{b}_i}$$

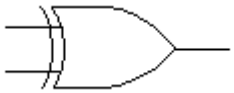
Mamy więc:

$$s_i = c_i(\overline{\bar{a}_i b_i + a_i\bar{b}_i}) + \bar{c}_i(\bar{a}_i b_i + a_i\bar{b}_i) = c_i(\overline{a_i \oplus b_i}) + \bar{c}_i(a_i \oplus b_i) = c_i \oplus (a_i \oplus b_i) = c_i \oplus a_i \oplus b_i$$

Wyrażenie $\bar{a}b + a\bar{b} = a \oplus b$ nazywamy „różnicą symetryczną” albo „sumą modulo dwa”. W tabelce przedstawiono wartości różnicy symetrycznej dla wszystkich argumentów.

b \ a	0	1
0	0	1
1	1	0

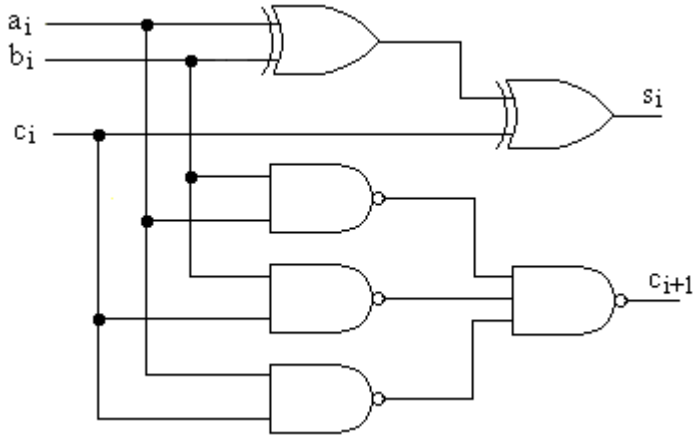
Funktor realizujący funkcję różnicy symetrycznej nazywany jest bramką XOR (inaczej: EX-OR). Bramka XOR bywa nazywana także ćwierćsumatorem.



Rys. 3 . Symbol bramki XOR.

Wzoru na wartość c_{i+1} nie da się przedstawić w postaci krótszej, niż ten wypisany na podst. tablicy Karnaugha.

Możemy narysować układ spełniający określone powyżej funkcje jednobitowego pełnego sumatora (posiadającego trzy wejścia dla trzech dodawanych cyfr):

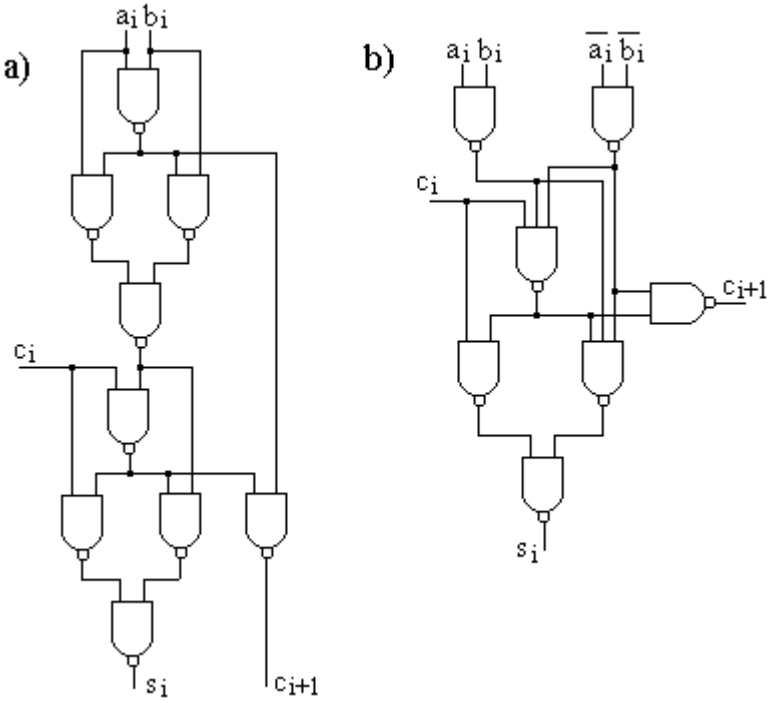


Rys. 4. Schemat dwójkowego sumatora elementarnego.

Każdy zbudowany sumator składa się z określonej liczby sumatorów elementarnych i w związku z tym jest przystosowany do dodawania liczb o określonej ilości cyfr (czyli - w naszym przypadku - bitów); ilość cyfr w dodawanych liczbach nie może być większa, niż ilość sumatorów elementarnych.
Podobne ograniczenia dotyczą układów logicznych realizujących operacje logiczne wewnątrz mikroprocesora. Z reguły wszystkie układy mikroprocesora są przystosowane do operowania na liczbach, czy - ogólniej mówiąc - słowach dwójkowych o tej samej długości. Tę wspólną długość nazywamy długością słowa mikroprocesora. Słowo 8-bitowe jest nazywane bajtem.

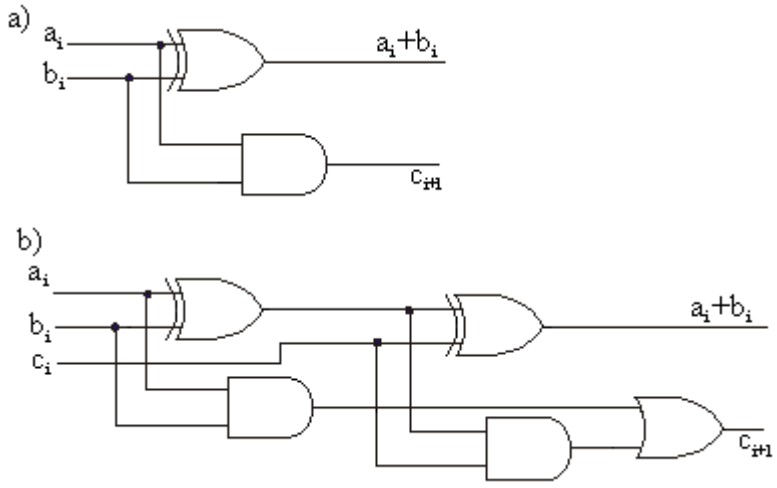
Na rysunku 5 zostały przedstawione schematy sumatorów jednobitowych zbudowanych z samych tylko bramek NAND. Rysunek 5b przedstawia realizację sumatora dla przypadku, w którym dysponujemy również zanegowanymi wartościami a_i oraz b_i (łatwo je otrzymać, stosując dwa elementy zaprzeczenia logicznego). Szybkość działania sumatorów wielobitowych, złożonych

z wielu sumatorów elementarnych z przeniesieniami szeregowymi (patrz rysunek 1) ograniczona jest opóźnieniami występującymi podczas transmisji przeniesień. Opóźnienia w takich układach dodają się. Budowane są więc układy transmitujące przeniesienia w sposób równoległy [1]. W sumatorze z przeniesieniami równoległymi wszystkie przeniesienia są wytwarzane jednocześnie na podstawie wartości odpowiednich bitów sumowanych składników i wartości przeniesienia początkowego.



Rys. 5. Realizacja sumatora jednobitowego z bramek NAND.

Na rysunku 6a przedstawiono realizację półsumatora z bramek XOR i AND. Półsumator nie posiada wejścia dla przeniesienia z poprzedniej pozycji, tzw. starego przeniesienia. Może on być wykorzystany jako sumator elementarny służący do dodawania cyfr występujących na najmniej znaczącej pozycji.



Rys. 6. Realizacja sumatora z dwóch półsumatorów : a) półsumator, b) sumator pełny.

Pełny sumator (rys. 6b) można zestawić z dwu półsumatorów uzupełnionych bramką OR

(równie dobrze może być to bramka XOR a to dlatego, że kombinacja stanów wejściowych, dla których te bramki różnią się, tutaj nie występuje).

W układach wielobitowych przeniesienie z i-tej pozycji: $C_{i+1} = A_iB_i + (A_i+B_i)C_i$
można podzielić na dwie części:
$A_iB_i = G_i$ - przeniesienie powstające na danej pozycji na podstawie wartości występujących tu cyfr sumowanych liczb,

$(A_i+B_i)C_i = T_iC_i$ - przeniesienie „transmitowane” z poprzedniej, mniej znaczącej pozycji .

Tak więc mamy: $C_{i+1} = G_i + T_iC_i$

Układ generujący przeniesienia można będzie opisać następującymi wyrażeniami:

$C_1 = G_0 + T_0C_0$

$C_2 = G_1 + T_1C_1 = G_1 + T_1(G_0 + T_0C_0) = G_1 + T_1G_0 + T_1T_0C_0$

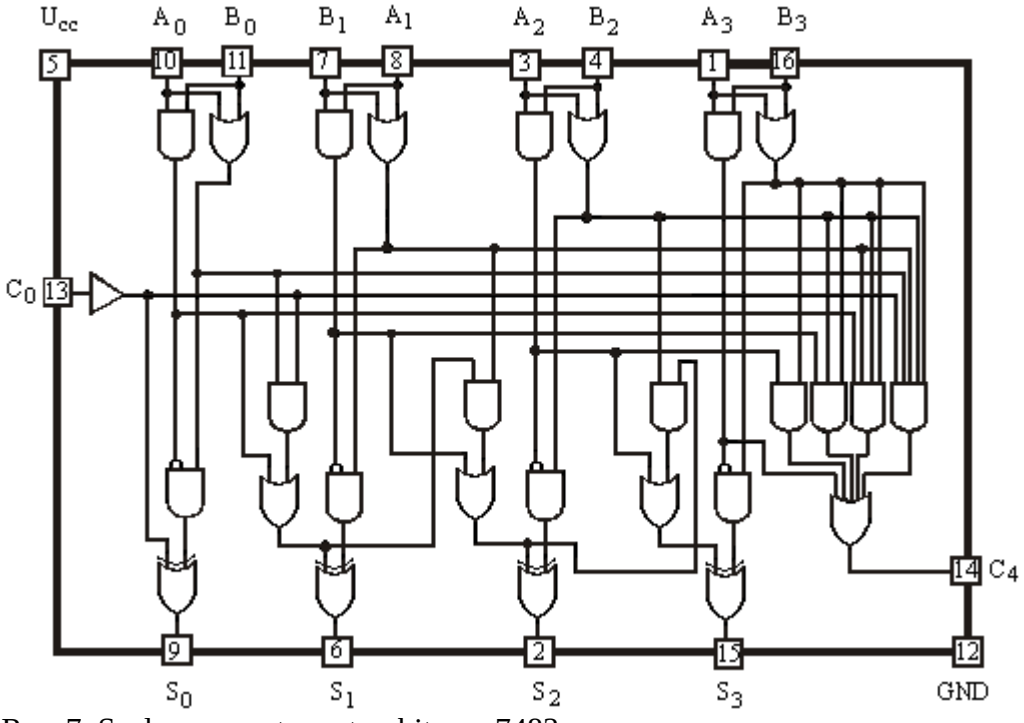
$C_3 = G_2 + T_2G_1 + T_2T_1G_0 + T_2T_1T_0C_0$

$C_4 = G_3 + T_3G_2 + T_3T_2G_1 + T_3T_2T_1G_0 + T_3T_2T_1T_0C_0$

$C_i = G_{i-1} + T_{i-1}G_{i-2} + T_{i-1}T_{i-2}G_{i-3} + ...+T_{i-1}T_{i-2}T_{i-3}.....T_0C_0$

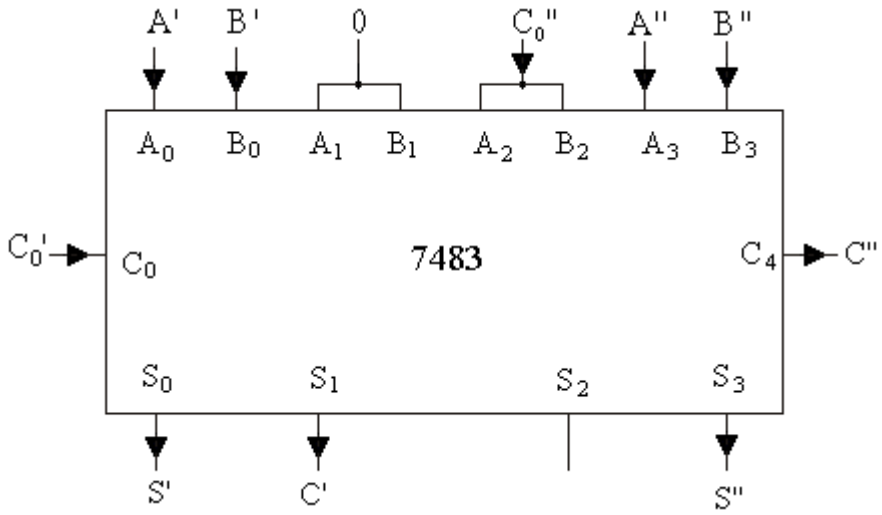
przy czym:
$G_i = A_iB_i;$ $T_i = A_i + B_i$

Powyższe prawidłowości dla przeniesień zostały wykorzystane przy projektowaniu 4-bitowego sumatora scalonego 7483. Sumator ten realizuje dodawanie dwu czterobitowych liczb binarnych i cyfry przeniesienia. Na rysunku 7 został pokazany schemat tego sumatora.



Rys. 7. Scalony sumator czterobitowy 7483.

Sumator czterobitowych liczb binarnych może być wykorzystany jako dwa niezależne sumatory liczb jednobitowych. Na rysunku 8 został przedstawiony sposób takiego wykorzystania sumatora 7483; na wejścia A_1 i B_1 są tu podawane na stałe zera, wyjście S_2 jest niewykorzystywane.



Rys. 8. Sposób utworzenia dwóch niezależnych sumatorów jednobitowych z sumatora 7483.

Kody cyfr dziesiętnych

Oprócz sumatorów, w których dane wejściowe i wynik operacji przedstawiony jest w kodzie binarnym, buduje się również takie, w których dane wejściowe i wynik sumowania przedstawiony jest liczbami dziesiętnymi, w których cyfry (od 0 do 9) są kodowane binarnie. Najczęściej stosowanymi kodami są: kod 8421 (tzw. kod BCD) oraz kod z nadmiarem 3. Oba te kody przedstawione są w tabeli:

Cyfry dziesiętne	Kod 8 4 2 1	Kod z nadmiarem 3
0	0 0 0 0	0 0 1 1
1	0 0 01	0 1 0 0
2	0 0 10	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

W kodzie BCD (od ang. Binary-Coded Decimal, czyli „dziesiętny zakodowany dwójkowo”) mamy zwykłe binarne kodowanie dziesięciu cyfr. Np. liczba 128, posiadająca w zapisie binarnym

postać 10000000, w kodzie BCD ma postać 0001 0010 1000, natomiast w kodzie z nadmiarem 3 ma postać 0100 0101 1011. Dla przejrzystości stosuje się spacje pomiędzy poszczególnymi przedstawieniami cyfr dziesiętnych. Jak widzimy, w kodach tych występują czteroelementowe grupy cyfr 0 i 1, tzw. tetrady, reprezentujące cyfry dziesiętne od 0 do 9.

Dodawanie liczb dziesiętnych wyrażanych w kodzie BCD.

Uwaga. Aby uniknąć nieporozumień, cyfry albo liczby dziesiętne będziemy tutaj oznaczać umieszczając (z prawej strony liczby albo cyfry) literkę „$_d$”. Dotyczy to liczb i cyfr złożonych tylko z zer i jedynek. Liczby albo cyfry zawierające znaki od 2 do 9, czyli dziesiętne, będziemy pisać bez dodatkowych oznaczeń.

Załóżmy, że sumator powinien poprawnie dodawać przedstawione w kodzie BCD dwie cyfry dziesiętne oraz przeniesienie z niższej pozycji. Wynik dodawania także powinien być przedstawiony w kodzie BCD. Ponieważ każda cyfra dziesiętna jest przedstawiana za pomocą czterech bitów, przeto sumator dziesiętny dodający dwie tak zakodowane cyfry oraz przeniesienie będzie składał się z sumatora dwójkowego 4–bitowego zawierającego 4 sumatory elementarne (np. przedstawione na rys. 4), połączone wg zasady występującej na rys.1 oraz z układu korekcyjnego, w którym czysto binarny wynik dodawania będzie zamieniany na poprawną postać BCD. Konieczność istnienia układu korekcyjnego wynika stąd, że największą dopuszczalną „cyfrą” w kodzie BCD jest 1001 (=9) i np. wynik 1100 (=12) powinien być zamieniony na wynik 0010 (=2) plus przeniesienie „1” do wyższej pozycji.

W sumatorze będą powstawać przeniesienia C_1, C_2, C_3, i C_4. Przeniesienie podawane z danego sumatora do sumatora dodającego dwie cyfry dziesiętne występujące na wyższej pozycji oznaczymy przez C. Gdy wynik dodawania przekracza wartość 9, wtedy powinno wystąpić przeniesienie C. Efekt zsumowania dwu cyfr – jak wiemy – składa się z wyniku i przeniesienia. Jeśli suma dodawanych cyfr nie przekracza wartości 9, sumator nasz ma pracować jak zwykły sumator binarny 4-pozycyjny – bez generowania przeniesienia do wyższej pozycji cyfry dziesiętnej.

Np.

```
 0100 (=4)
+0101 (=5)
=1001  (=9; nie wystąpiło przeniesienie; wynik jest tu równy sumie binarnej)
```

Tu wynik dodawania wynosi 9, więc na wyjściu sumatora powinna pojawić się suma binarna cyfr, bez przeniesienia C.

Gdy wartość sumy cyfr przekracza 9, układ korekcyjny powinien wygenerować przeniesienie C oraz skorygować wynik tak, aby suma przeniesienia C i wyniku skorygowanego była równa sumie dodawanych cyfr. Ponieważ wartość przeniesienia C wynosząca 10_d jest o 6 mniejsza, niż wartość przeniesienia C_4, wynosząca 16, więc wartość wyniku powinna zostać powiększona o 6. Gdyby nie skorygować wyniku dodania do siebie dwu cyfr, np. 1000+1000 (= 8+8), to otrzymana binarnie suma mająca postać 00010000 w kodzie BCD posiada wartość 10_d a nie 16. Tak więc korekcja wyniku powinna polegać na zwiększeniu jego wartości o 6, czyli binarnie o 0110.

Przekroczenie przez sumę cyfr liczby 9 można rozpoznać:
- dla sumy zawierającej się w granicach od 10_d do 15 włącznie - po miejscu wystąpienia jedynek w wyniku,
- dla sumy większej od 15 - po wystąpieniu przeniesienia C_4 .

Oznaczmy cyfry wyniku – poczynając od najmniej znaczącej pozycji - przez U_0, U_1, U_2, U_3. Jeśli na pozycji U_3 i jednocześnie na pozycji U_2 lub U_1 występują jedynki, wtedy wartość sumy zawiera się w granicach od 10_d do 15 (i powinno zostać wygenerowane przeniesienie).

Przykład 1.

```
      0111  (=7)
    + 0011  (=3)
1<- = 1010  (suma binarna; przen. C4 nie wystąpiło; U3=1 oraz U1=1, więc przen. C wystąpi)
    + 0110  (korekcja wyniku)
1<- = 0000  (przeniesienie C o wartości 10d oraz wynik o wartości 0 dają sumę 10d; przenie-
             sienie powstające podczas korekcji wyniku jest ignorowane)
```

Tu wynik dodawania wynosi 10_d i na pozycjach U_3 i U_1 występują jedynki, więc powinno wystąpić przeniesienie C (oznaczone symbolicznie jako „1<-„) oraz zwiększenie wyniku o wartość 0110. Przeniesienie C_4 powstające podczas korekcji wyniku jest pomijane.

Przykład 2

```
       1001 (=9)
     + 1001 (=9)
= 1<-  0010  suma binarna; wystąpiło przeniesienie: C4=1=C)
     + 0110 (korekcja wyniku)
= 1<-  1000 (=8; przeniesienie C o wartości 10d oraz wynik 8 dają sumę 18)
```

Tu wystąpiło przeniesienie C_4 jeszcze przed korekcją wyniku, czyli wartość sumy cyfr przekracza 9 więc powinno wystąpić przeniesienie C oraz zwiększenie wyniku o 0110.

Sumatory dziesiętne.

W obu powyższych przykładach brak jest przeniesienia z niższej pozycji. W sumatorze pełnym suma binarna jest wynikiem dodania nie dwu, ale trzech cyfr: dwu cyfr występujących na danej pozycji w obu sumowanych liczbach oraz przeniesienia z niższej pozycji. Jeśli wartość właśnie tak otrzymanej sumy binarnej przekracza 9, układ korekcyjny powinien wygenerować przeniesienie oraz skorygować wynik.

Boolowski wzór na przeniesienie dziesiętne w sumatorze posiadającym wejście dla przeniesienia z niższej pozycji wyrazi się wzorem:

$$C = C_4 + U_3(U_2 + U_1)$$

Jeśli C=1, to powinno wystąpić przeniesienie oraz korekcja wyniku.

Po przekształceniu mamy:

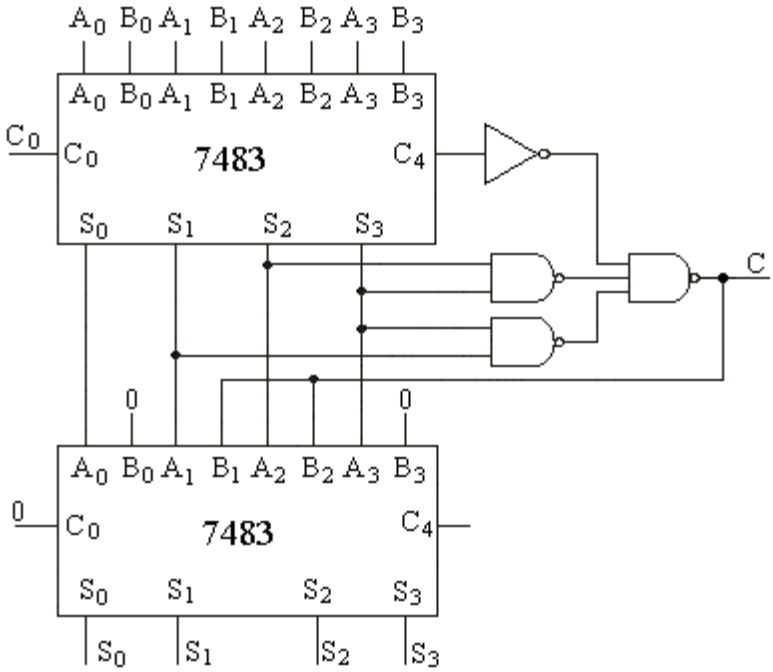
$$C = C_4 + U_2U_3 + U_1U_3$$

Zastępując sumę zaprzeczeniem iloczynu zaprzeczeń otrzymujemy:

$$C = \overline{\overline{C_4}\,\overline{U_2U_3}\,\overline{U_1U_3}}$$

Wzór powyższy znalazł praktyczną realizację w układzie przedstawionym na rys. 9. Odpowiednio

połączone dwa sumatory binarne 7483 oraz kilka elementów logicznych tworzą tu sumator pełny dodający dwie cyfry dziesiętne kodowane binarnie.



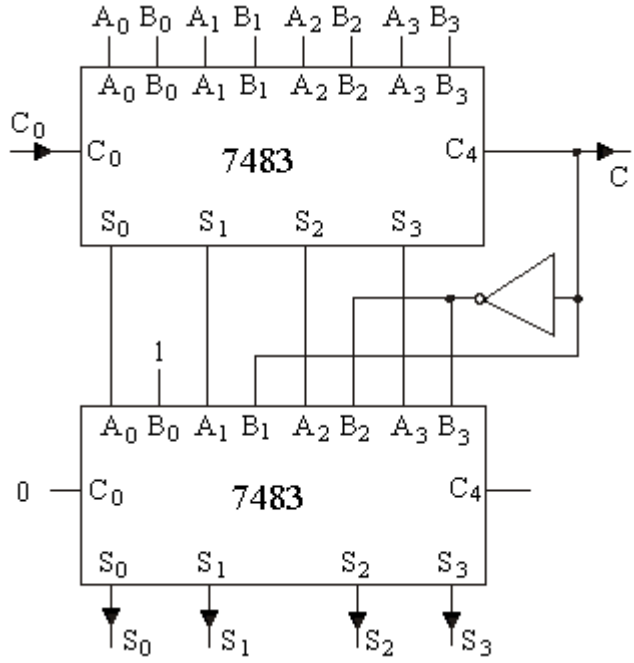
Rys. 9. Schemat sumatora dziesiętnego dodającego liczby wyrażone w kodzie BCD.

Układ górny 7483 jest sumatorem dwu liczb binarnych (liczby $A_3\ A_2\ A_1\ A_0$ i liczby $B_3\ B_2\ B_1\ B_0$) oraz przeniesienia C_0. Trzy bramki NAND oraz element zaprzeczenia tworzą układ podający na wyjście C (przeniesienie) jedynkę logiczną, gdy suma liczb $A_3\ A_2\ A_1\ A_0$, $B_3\ B_2\ B_1\ B_0$ i przeniesienia C_0 jest większa, niż 9. Dolny układ 7483 jest sumatorem korygującym wynik. Jeśli C=0, to do wyniku podawanego z sumatora górnego do sumatora dolnego jest dodawana liczba binarna 0000 i wynik nie ulega zmianie. Jeśli C=1, wtedy następuje korekcja wyniku: do powstałego w sumatorze górnym wyniku dodawana jest w sumatorze dolnym liczba 0110 ($=6_d$); na wyjściu sumatora dolnego (przewody $S_3\ S_2\ S_1\ S_0$) mamy wynik skorygowany.

Kod z nadmiarem 3 został rozpowszechniony, gdyż jest on wygodny w tworzeniu układów odejmujących liczby dziesiętne.

Otóż kod ten ma szczególną właściwość: zaprzeczając bity dowolnej cyfry dziesiętnej wyrażonej w tym kodzie, otrzymujemy cyfrę będącą jej dopełnieniem do 9; np.: cyfra 0111 (posiadająca wartość 4) oraz cyfra o bitach zaprzeczonych 1000 (posiadająca wartość 5) dają po zsumowaniu wartość 9, cyfra 1001 (posiadająca wartość 6) oraz cyfra o bitach zaprzeczonych 0110 (posiadająca wartość 3) dają po zsumowaniu wartość 9 itd. Obliczenie różnicy dwu cyfr dziesiętnych polega na dodaniu do odjemnej dopełnienia (do dziesięciu) odjemnika i – gdy w wyniku tego dodawania nie wystąpi przeniesienie – odjęciu od cyfry znajdującej się na wyższej pozycji tzw. pożyczki (czyli cyfry 1). Aby otrzymać dopełnienie do 10, należy otrzymane (poprzez zaprzeczenie bitów) dopełnienie zmienić tak, by jego wartość stała się większa o 1. Łatwo jest to zrealizować dodając do cyfry otrzymanej poprzez zaprzeczenie bitów (właściwiej będzie tu powiedzieć: dodając do liczby otrzymanej poprzez zaprzeczenie bitów) liczbę 0001 w zwykłym sumatorze binarnym. Odejmowaniem liczb nie będziemy się tu zajmować szczegółowo.

Sumator w kodzie z nadmiarem 3 przedstawiony na rys. 10 działa w ten sposób, że – w zależności od wystąpienia w sumatorze górnym przeniesienia C_4 - do wyniku otrzymanego w tymże sumatorze jest dodawana w sumatorze dolnym określona liczba. Gdy w sumatorze górnym wystąpi przeniesienie C_4 (tzn. gdy C_4=1; zajdzie to wtedy, gdy suma wartości cyfr przekroczy wartość 9), to do wyniku zostanie dodane binarnie liczba 0011. Gdy tego przeniesienia nie ma (tzn. gdy C_4=0), to od wyniku należałoby odjąć binarnie liczbę 0011. Okazuje się, że zamiast odejmować liczbę 0011, można binarnie dodać do wyniku liczbę 1101 (powstałe ewentualnie przeniesienie zignorować) i wynik końcowy będzie ten sam. Występujący na wyjściu sumatora dolnego (przewody S_0, S_1, S_2 i S_3) wynik jest przedstawiony w kodzie z nadmiarem 3.



Rys. 10. Schemat blokowy sumatora dziesiętnego pracującego w kodzie z nadmiarem 3.

W układzie przedstawionym na rys. 10 mamy: C = C_4 (chodzi tu o wyjście C_4 sumatora górnego; w układach z rys. 9 i 10 przeniesienia C_4 powstające w sumatorach dolnych są ignorowane). Kod z nadmiarem 3 ma w tym układzie korzystną właściwość polegającą na tym, że jeśli łączna suma dodawanych cyfr i przeniesienia C_0 przekracza wartość 9, to w sumatorze (górnym) wystąpi przeniesienie; nie potrzeba więc tu dodatkowego układu analizującego wartość sumy, jak to było w układzie z rys. 9.

Przykład 1.

```
 0100 (=1)
+0101 (=2)
=1001 (suma binarna, brak przeniesienia C4)
+1101 (korekcja)
=0110 (=3; przeniesienie powstające podczas korekcji jest ignorowane)
```

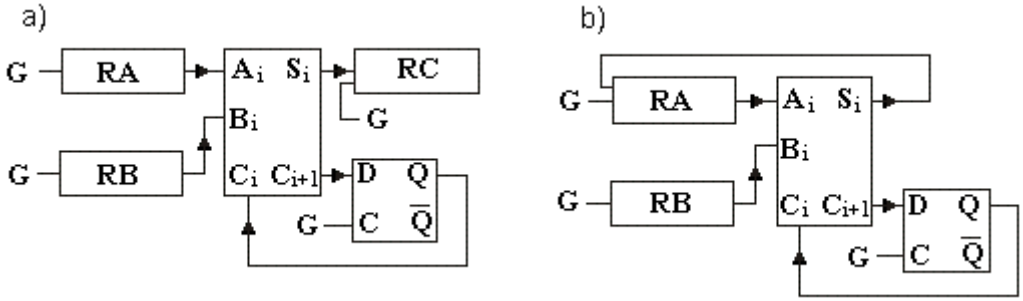
Przykład 2.

```
   1000 (=5)
  +1001 (=6)
=1<- 0001 (suma binarna, wystąpiło przeniesienie C4 w sumatorze górnym: C4=1=C)
  +0011 (korekcja wyniku)
=1<- 0100 (=1; przeniesienie C o wartości 10d oraz wynik 1 dają sumę 11d)
```

Oprócz sumatorów równoległych, zbudowanych z wielu sumatorów elementarnych dodających cyfry znajdujące się na takich samych pozycjach, stosowane są sumatory szeregowe zawierające jeden sumator elementarny, rejestry przesuwające podające cyfry dodawanych liczb, rejestr przechowujący powstałe przeniesienie oraz rejestr przesuwający odbierający sumę. Rejestrem przechowującym przeniesienie może być przerzutnik D. Przerzutnik oraz rejestry są taktowane sygnałem z generatora (na rys. 11 symbolem G zaznaczono wejścia rejestrów, do których ma być doprowadzony sygnał z generatora). Sygnał z generatora przesuwa zawartości rejestrów przesuwających.

Rysunek 11a przedstawia schemat blokowy sumatora szeregowego zwykłego. Do dwu wejść A_i i B_i sumatora szeregowego z dwu rejestrów przesuwających RA i RB podawane są kolejne pary cyfr znajdujących się na tej samej pozycji (poczynając od pozycji najniższej) dodawanych liczb. Sumator dodaje występujące aktualnie na wejściach A_i i B_i cyfry i podawane z przerzutnika przeniesienie. Powstały wynik dodania cyfr i przeniesienia jest odbierany przez rejestr przesuwający RC. Powstałe przeniesienie jest odbierane przez przerzutnik; w następnym takcie przeniesienie to zostanie podane na wejście C_i sumatora elementarnego a na wejścia A_i i B_i zostanie podana kolejna para cyfr dodawanych liczb.

W sumatorze przedstawionym na rys. 11b powstający wynik jest wprowadzany do rejestru jednego ze składników. Występująca na danej pozycji cyfra jest zamieniana przez cyfrę wyniku. Sumator taki nosi nazwę sumatora akumulującego, gdyż gromadzi w rejestrze akumulującym RA (akumulatorze) sumy kolejnych liczb pobieranych z rejestru RB.



Rys. 11. Sumator szeregowy a) zwykły, b) akumulujący.

Zadania.

1. W oparciu o rysunek 6a zbudować półsumator. Przeanalizować jego działanie. Z dwóch półsumatorów zbudować i zbadać pełny sumator (patrz rysunek 6b). Przedstawić w tabelkach wyniki działania układów dla wszystkich możliwych kombinacji stanów wejściowych.

2. Zbudować w oparciu o rysunek 4 jednobitowy sumator z bramek NAND, przeanalizować działanie i przedstawić tabelę stanów dla wszystkich możliwych kombinacji stanów wejściowych.

3. Zbadać sumator 4-bitowy zbudowany w oparciu o układ scalony 7483. Przedstawić w tabelkach

wyniki sumowania dwu liczb: nie dających przeniesienia C_4 oraz dających przeniesienie C_4 przy C_0=0, następnie dla tych samych liczb przy C=1 (4 sumowania).

4. Utwórz z układu 7483 dwa niezależne sumatory jednobitowe (patrz rysunek 8). Sprawdź działanie obu sumatorów i podaj w tabelkach przykładowe wyniki działania obu – po 3 przykłady dla każdego; niech dla obu sumatorów wystąpią rozmaite przypadki: C_0=0, C_0=1, C=0 oraz C=1.

5. Zestawić i zbadać sumator dziesiętny w kodzie 8421 według schematu przedstawionego na rysunku 9. W tabelkach przedstawić wyniki działania tego sumatora:

- dla dwu liczb dodawanych, których suma nie przekracza 9 przy C_0=0,
- dla dwu liczb dodawanych, których suma wynosi 9 przy C_0=1 (ma wystąpić C=1),
- dla dwu liczb , których suma zawiera się w przedziale od 10 do15 przy C_0=0,
- dla dwu liczb, których suma zawiera się w przedziale od 16 do 18 przy C_0=0 albo C_0=1.

6. Zestawić i zbadać sumator dziesiętny w kodzie z nadmiarem 3 (schemat według rysunku 10). Przedstawić w tabelkach dwa przykłady sumowania dwu liczb, tak by w przykładach wystąpiły C_0=0, C_0=1, C=0 oraz C=1.

Wykaz literatury:

1. P. Misiurewicz, M. Grzybek, Półprzewodnikowe układy logiczne TTL, Wydawnictwa Naukowo-Techniczne, Warszawa 1979

2. A. Rydzewski, K. Sacha, Mikrokomputer - elementy, budowa, działanie, Wydawnictwo NOT SIGMA, Warszawa 1987

Opracowanie uzupełnił Roman Kazański. Lublin, 21 sierpnia 2008r. Ostatnia zmiana 17 kwietnia 2009r. Plik sumatork.doc