ДВОИЧНО-ДЕСЕТИЧЕН СУМАТОР В ТЕГЛОВЕН КОД 5211

Димитър С. Тянев

Резюме: В статията е изложен структурният синтез на двоично десетичен суматор в тегловен код 5211. Кодът 5211 е актуален във връзка с развиващата се десетична хардуерно реализирана аритметика, имаща за цел да избегне известните проблеми на двоичната. Синтезът е представен в два варианта за две от възможните 64 на брой кодови таблици. За всяка от кодовите таблици, освен операция събиране, е разгледана и операция изваждане. Синтезирани са съответно два варианта на логическа структура на устройство за събиране и изваждане на цели числа, представени в допълнителен код. Приведени са числени примери илюстриращи формулираните правила и функционирането на логическите структури за всяка от кодовите таблици.

BCD-adder in a weight code 5211

Dimitar S. Tyanev

Abstract: The structure synthesis of binary coded decimal adder in a weighted code 5211 is presented. This code is a question of present interest because of the developing of the hardware implemented decimal arithmetic, which basic goal is to avoid the known problems of the binary. The synthesis is presented in two versions for two of the 64 code tables. Both addition and subtraction operations are considered for each code table. Two logical structures for addition and subtraction of integers represented in complementary code are synthesized. Numerical examples are described. They illustrate the formulated rules and the functioning of logical structures for each of the code tables.

1. Въведение

Използването на двоична аритметика води до минимизиране на хардуера в компютърните системи. Това се случва в алгоритмично, структурно и технологично отношение и се дължи на свойството дуалност, свързващо Булевата логика с двоичната бройна система [1], [2]. Различията между представяните данни и хардуера се контролира и компенсира с помощта на софтуер. Софтуерната поддръжка на десетична аритметика, както и софтуерно изпълняваните преобразования между двоична и десетична бройна система с цел визуализация, са с висока честота, което води до огромни загуби на време и до снижаване на производителността. Днес компютърните системи включват в минимална степен десетичен хардуер. Търговската, финансовата, комуникационната, застрахователната и научната дейности обаче, които се основават на десетични данни, се нуждаят от бърз напредък в технологията, подкрепяща десетичната аритметика в компютърната техника. Вече са натрупани наблюдения, които показват, че много приложения извършват десетична обработка от 50% до 90% на времето [8]. Необходимостта от десетична аритметика в хардуера е спешна. Ето защо тя се подкрепя от стандарта IEEE 754/2008 [5], [6], [7], [8]. Компанията IBM от 2007 година включва в процесора си Z9 устройство за работа с десетична плаваща запетая (DFP) [9]. Десетичната аритметика е още по-застъпена в следващата генерация Z10 както за числа с фиксирана, така и за числа с плаваща запетая [10]. Процесори IBM Power6, 7 също съдържат DFP устройства. Операционната система на IBM Z/OS, компилаторите за Java, C/C++ както и СУБД DB2 на компания IBM поддържат този десетичен хардуер [8].

Десетичните аритметични устройства по своята същност са по-сложни, от двоичните, тъй като те трябва да се справят с повече различни цифри. Не трябва да се забравят и 6-те излишни кодови комбинации, които създават затруднения в синтеза на хардуера.

Основен недостатък на двоичната аритметика са неточните решения при използване на десетични фракции на числата, т.е. на дробните части на числата. Източниците за тези неточности при използване на двоичното представяне на числата във формата с плаваща запетая са много [1]. Този недостатък е отлично илюстриран с примери в [1]. [2], [3], [4], [5], [6], [7], [8].

Кодирането на десетични цифри е ключов въпрос за осигуряване на бърза десетична аритметика [12], [14]. Последните разработки предполага използването на алтернативни кодове BCD кодове, като например на BCD-5211, чрез който при събиране се постига пълна аналогия с десетичния пренос. Изборът на код за представяне на операнди е пряко свързан със стойността на грешката при конкретното форматиране на фракцията. Кодът BCD-5211 предоставя определено предимство по отношение на код BCD-4221 или BCD-8421.

Кодът 5211 е тегловно значим и е един от 17-те 4-битови BCD-кода с положителни тегла [13]. В дисертацията [14] този код се определя като перспективен за реализация на десетична аритметика, особено за операции умножение и деление. Този код е актуален, ето защо е обект на настоящото изследване. При операции събиране и изваждане, както ще бъде показано по-долу, имитира десетичния пренос (съответно заем).

2. BCD тегловен код 5211

При кодиране на десетичните цифри в код 5211 се използват само 10 от възможните 16 4-битови двоични комбинации. Еднозначно се кодират само цифрите 0, 4, 5 и 9. Останалите шест цифри не могат да се кодират еднозначно. За всяка от цифрите 1, 2, 3, 6, 7 и 8 могат да се формират по две различни кодови комбинации, така че общият брой на възможните кодови таблици е 2^6=64. Този брой е твърде голям за да е възможно изследването му тук. В литературните източници не се намира единно предложение за избор на кодова таблица. Авторите обикновено залагат на свойството допълняемост [1], [2]. Нашата проверка на кодова таблица с това свойство обаче показа, че то води до по-сложни правила за операция събиране. Изследвани са следните две кодови таблици

Кодови таблици на 2/10-чен тегловен код 5211 ($t_3t_2t_1t_0$)

Таблица 2.1

d	t_3	t_2	t_1	t_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	1
8	1	1	0	1
9	1	1	1	1

Таблица 2.2

d	t_3	t_2	t_1	t_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	0	1
9	1	1	1	1

3. Правила за 2/10-чно събиране в код 5211 (кодова таблица 2.1)

Както при всеки известен 2/10-чен суматор [2] и тук за операция събиране се налага синтез на двуетажна логическа структура на суматора в този код. Това се дължи на различията между бройните системи. Както на първия, така и на втория суматор се използват 4-битови двоични суматори. Двоичната сумата от първия етаж $z_i^{'} = x_i + y_i + p_{i-1}$ не винаги е вярна (в смисъла на кодовата таблица). Във всички случаи, когато тя не е вярна, се налага корекция, стойността на която се определя по силата на синтезираните и представени по-долу правила. В резултат на тях сумата от втория етаж $z_i = z_i^{'} + K_i$ както и генерираният тетраден пренос p_i трябва да съответстват по стойност на десетичните.

Синтезираните правила са разделени на две групи

1. Тетрадни суми без входящ пренос p_{i-1};
2. Тетрадни суми с входящ пренос p_{i-1}.

Първа група правила

Когато получената на първия етаж сума е в съответствие с кодовата таблица 2.1, тя не се нуждае от корекция. За получаваните на първия етаж суми $z_i^{'}$, на които не съответстват кодови комбинации, са синтезирани правила за корекция, сведени в таблица 3.1.

Таблица 3.1. Корекции на суми без входящ пренос $z_i^{'} = x_i + y_i$

Забранена комбинация	Корекция при липса на пренос от старшия бит на тетрадата $p_3^{'} = 0$	Корекция при наличие на пренос от старшия бит на тетрадата $p_3^{'} = 1$
0 0 1 0	+1	(-1) или (+1) ако $x_3 \cap y_3 = 1$
0 1 0 0	+1	(-1) или (+1) ако $x_3 \cap y_3 = 1$
0 1 1 0	+1	(-1) или (+1) ако $x_3 \cap y_3 = 1$
1 0 1 0	(+1) или (-1) ако $\overline{x_3} \cap \overline{y_3} = 1$	-1
1 1 0 0	(+1) или (-1) ако $\overline{x_3} \cap \overline{y_3} = 1$	-1
1 1 1 0	(+1) или (-1) ако $\overline{x_3} \cap \overline{y_3} = 1$	-1

В случаи на корекция на тетрадната сума с (-1), възникналият на втория етаж пренос, не се разпространява към следващата по-старша тетрада.

Втора група правила

В този случай към двете входни комбинации се добавя пренос от по-младшия разряд $p_{i-1} = 1$, т.е. $z_i^{'} = x_i + y_i + p_{i-1}$. Правилата за корекция за суми, получени на първия етаж, представени в таблица 3.1, са в сила и за тази група.

В случай, че получената на първия етаж тетрадна сума е в съответствие с кодова таблица 2.1, тя не се нуждае от корекция. Последното обаче не винаги е вярно, тъй като има случаи, в които сумата е вярна според кодовата таблица, но не е вярна от гледна точка на крайния резултат. Ето защо се налага нова корекция, която да формира крайния резултат в съответствие с десетичния. Тя се извършва в суматора на втория етаж. Според синтезираните за група правила, корекциите в отделните ситуации са две: (+1) или (+2). Конкретните правила са сведени в таблица 3.2.

Таблица 3.2. Корекции на суми с входящ пренос $z_i^{'} = x_i + y_i + p_{i-1}$

Корекция	Разпознаващо правило
+1	Когато получената тетрадна сума на първия етаж е (0111). Комбинацията е в съответствие, но крайната сума не е вярна.
+1	Когато получената тетрадна сума на първия етаж е (1111), но липсва тетраден пренос, т.е. когато $p_i = 0$.
+2	Когато получената тетрадна сума на първия етаж е (0011) и заедно с това старшите битове на двете входни комбинации са еднакви, т.е. ($x_3 = y_3 = 0$) или ($x_3 = y_3 = 1$).
+2	Когато получената тетрадна сума на първия етаж е (0101) и заедно с това старшите битове на двете входни комбинации са еднакви, т.е. ($x_3 = y_3 = 0$) или ($x_3 = y_3 = 1$).
+2	Когато получената тетрадна сума на първия етаж е (1011) или (1101) и заедно с това старшите битове на двете входни комбинации са различни, т.е. ($x_3 \neq y_3$).

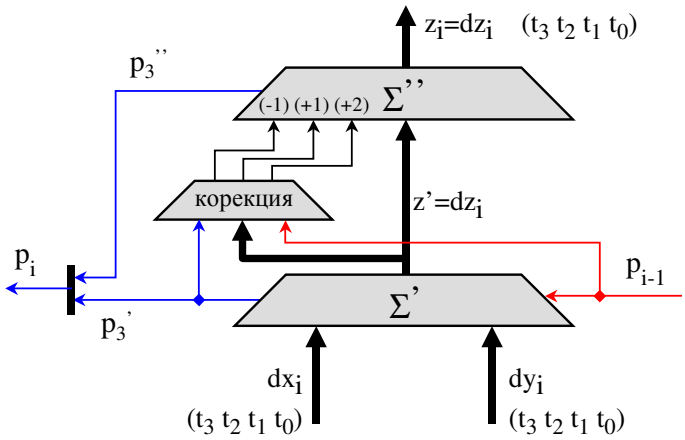
Във всички останали случаи корекция няма.

Пример

```
  1   1 1 1 1
  0 7 2 6 2 8 7
                +
  0 8 1 6 7 6 9
---------------
  1 5 4 3 0 5 6
```

```
    1       1   1       1   1       1
 0000    1011    0011    1001    0011    1101    1011
                                                     +
 0000    1101    0001    1001    1011    1001    1111
-------------------------------------------------------
 0001    1000    0101    0011    1111    0111    1010
                                                     +
                 10      10      1       1       1111
-------------------------------------------------------
 0001    1000    0111    0101    0000    1000    1001
```

От по-горе изложеното, аналогично на вече известните за други BCD кодове [1], [2], така и тук за код 5211, се прави извод, че логическата структура на едноразрядния суматор в код 5211, ще съдържа два двоични 4-битови суматора и схема за генериране на корекциите. Тази логическа структура има вида, показан на фиг.3.1.



Фиг. 3.1. *Логическа структура на едноразряден BCD суматор в код 5211 (таблица 2.1)*

4. Схема на допълнителен код за кодова таблица 2.1

За изпълнение на операция изваждане е необходимо числата със знак да бъдат представени в допълнителен код [1]. Допълнителният код на отрицателните числа може да се дефинира чрез помощния обратен код, като сума

$$[X]_{ДК} = [X]_{ОК} + 1. \tag{4.1}$$

Тъй като разглеждания тук тегловен код не притежава свойството допълняемост [1], необходимо да бъде синтезирана специална схема за неговото получаване, която се основава на определението за обратен код

$$[X]_{ОК} = \begin{cases} X, & X > 0; \\ 10^n - 1, & X < 0. \end{cases} \tag{4.2}$$

Определение (1) се постига поразрядно чрез разликата

$$(9 - d_k), \tag{4.3}$$

където с d_k е означена десетичната цифра на числото X, стояща в k-ия разряд. Така в таблица са изразени допълненията на десетичните цифри до 9, в съответствие с тяхната кодова таблица 2.1.

Таблица 4.1. Дефиниране на обратен код

Цифра d_k	Допълнение до 9	Код на цифрата $t_3t_2t_1t_0$	Инверсия на кода	Корекция	Код на допълнението
0	9	0000	1111		1111
1	8	0001	1110	-1	1101
2	7	0011	1100	-1	1011
3	6	0101	1010	-1	1001
4	5	0111	1000		1000
5	4	1000	0111		0111
6	3	1001	0110	-1	0101
7	2	1011	0100	-1	0011
8	1	1101	0010	-1	0001
9	0	1111	0000		0000

Както се вижда кодът на допълнението, т.е. разликата (3), може да се получи след побитова инверсия на кодовата комбинация на съответната десетична цифра и корекция (-1), ако е необходимо. Тъй като корекцията трябва да се постига автоматично, за целта трябва да се синтезира логическа функция за корекция. Таблица 4.1 се разглежда като таблица на истинност, от която следва представената карта на Карно (фиг.4.1) и логическото уравнение за функцията на корекция с (-1).

t3 t2 \ t1 t0	00	01	11	10
00	0	*	*	1
01	1	*	0	1
11	1	*	0	1
10	0	*	*	1

Фиг. 4.1. *Карта на Карно за функцията на корекция с* (-1)

$$K\underset{(-1)}{2.1} = (t_2 \cap \overline{t_1}) \cup (t_1 \cap \overline{t_0}) \ . \qquad (4.4)$$

Пример 4.1. Да се представи в допълнителен код числото X = -574906

$$[X]_{ПК} = 1\ \ 1000\ \ 1011\ \ 0111\ \ 1111\ \ 0000\ \ 1001$$

Обратният код на това число се получава от правия след побитова инверсия и корекция в съответните тетради с (-1).

1	1000	1011	0111	1111	0000	1001	
1	0111	0100	1000	0000	1111	0110	¬
0		1111				1111	+
1	0111	0011	1000	0000	1111	0101	

Допълнителният код се получава след прибавяне на единица към обратния код.

1	0111	0011	1000	0000	1111	0101	+
						0001	
1	0111	0011	1000	0000	1111	0110	+
						1	
1	0111	0011	1000	0000	1111	0111	

$$[X]_{ДК} = 1\ \ 0111\ \ 0011\ \ 1000\ \ 0000\ \ 1111\ \ 0111$$

Пример 4.2. Да се изпълни операция събиране Z=X+Y, където X = -574906, Y = 297318. Резултатът трябва да бъде Z = -277588, получен в допълнителен код.

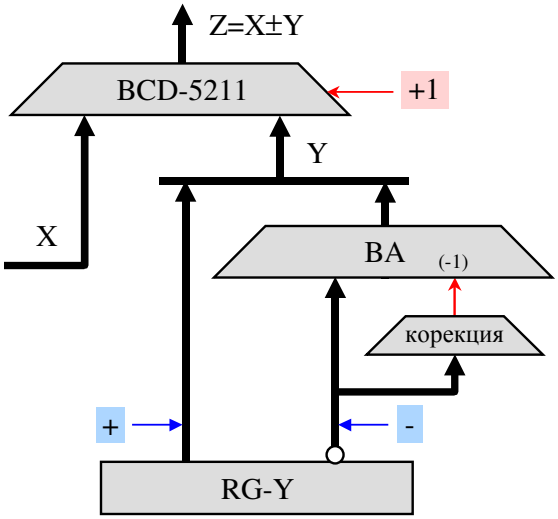
	1	1		1	1		
1	0111	0011	1000	0000	1111	0111	$= [X]_{дк}$
						+	
0	0011	1111	1011	0101	0001	1101	$= [Y]_{дк}$
1	1011	0011	0011	0110	0001	0100	
						+	
0				1		1111	
1	1011	0011	0011	0111	0001	0100	$= [Z]_{дк}$

Проверка:

1	1011	0011	0011	0111	0001	0100	$= [Z]_{дк}$
						¬	
1	0100	1100	1100	1000	1110	1011	
						+	
0	1111	1111	1111		1111		
1	0011	1011	1011	1000	1101	1011	
						+	
						0001	
1	0011	1011	1011	1000	1101	1100	
						+	
						1	
1	0011	1011	1011	1000	1101	1101	$= [Z]_{пк}$

Z = -277588

Апаратната реализация на схемата за преобразуване на числа със знак от прав код в допълнителен може лесно да бъде синтезирана въз основа на демонстрираните по-горе правила. На фиг.4.2 е представена логическа структура на устройство за събиране и изваждане в допълнителен код на BCD-числа, цифрите на които са кодирани в код 5211 в съответствие с кодова таблица 2.1.



Фиг. 4.2. *Логическа структура на многоразряден* 2/10*-чен суматор за допълнителен код в BCD код 5211 (таблица 2.1)*

Както може да се види, инверсните стойности на битовете на всяка от тетрадите на едно многоразрядно BCD-число Y, се вземат от инверсните изходи на тригерите в регистъра. За събиране с необходимите поразрядни корекции (-1) е включен двоичен сума-

тор ВА (*Binary adder*). В този суматор липсват вериги за разпространение на тетрадни преноси, което се налага от използвания допълнителен код за представяне на корекцията (-1). Практически той е съставен от толкова 4-битови двоични суматори, колкото е дължината на десетичната разрядна мрежа.

За формиране на тетрадните корекции (-1) са включени дешифратори, реализиращи логическа функция (4.4), синтезирана в съответствие с таблица 4.1. Добавянето на единица за окончателно формиране на допълнителния код в съответствие с (1), е постигнато по линия на преноса в младшия разряд на BCD-суматора.

5. Правила за 2/10-чно събиране в код 5211 (кодова таблица 2.2)

И тук за операция събиране се налага синтез на суматор с двуетажна логическа структура. Двоичната сумата от първия етаж $z_i^{'} = x_i + y_i + p_{i-1}$ не винаги е вярна (в смисъла на кодовата таблица). Във всички случаи, когато тя не е вярна, се налага корекция, стойността на която се определя по силата на представени по-долу правила. В резултат на тях сумата от втория етаж $z_i = z_i^{'} + K_i$ както и генерираният тетраден пренос p_i трябва да съответстват по стойност на десетичните. И тук синтезираните правила са разделени на две групи:

1. Тетрадни суми без входящ пренос p_{i-1};
2. Тетрадни суми с входящ пренос p_{i-1}.

Първа група правила

В случай, че получената на първия етаж тетрадна сума е в съответствие с кодовата таблица 2.2, тя не се нуждае от корекция, с две изключения. Изключенията са вписани в таблица 5.1. За получаваните на първия етаж суми $z_i^{'}$, на които не съответстват кодови комбинации, са синтезирани правила за корекция, сведени в таблица 5.1.

Вписаните в последните два реда комбинации не са забранени, но не съответстват на правилната сума. В случаи на корекция на тетрадната сума с (-1), възникналият на втория етаж пренос, не се разпространява към следващата по-старша тетрада.

Втора група правила

В този случай към двете входни комбинации се добавя пренос от по-младшия разряд $p_{i-1} = 1$, т.е. $z_i^{'} = x_i + y_i + p_{i-1}$.

Таблица 5.1. Корекции на суми без входящ пренос $z_i^{'} = x_i + y_i$

Забранена комбинация	Корекция при липса на пренос от старшия бит на тетрадата $p_3^{'} = 0$	Корекция при наличие на пренос от старшия бит на тетрадата $p_3^{'} = 1$
0 0 1 0	+1	-1
0 1 0 0	+1	-1
0 1 1 0	+1	-1
1 0 0 1	+1	-1
1 0 1 1	+1	-1
1 1 1 0	-1	
0 0 0 1		-1
0 0 1 1		-2

Правилата за корекция за суми, получени на първия етаж, представени в таблица 5.1, са в сила и за тази група.

В случай, че получената на първия етаж тетрадна сума е в съответствие с кодова таблица 2.2, тя не се нуждае от корекция. Последното обаче не винаги е вярно, тъй като има случаи, в които сумата е вярна според кодовата таблица, но не е вярна от гледна точка на крайния резултат. Ето защо се налага нова корекция, която да формира крайния резултат в съответствие с десетичния. Тя се извършва в суматора на втория етаж. Корекциите в отделните ситуации са три: (+1), (-1) или (+2). Тези конкретни ситуации правилата са сведени в таблица 5.2.

Таблица 5.2. Корекции на суми с входящ пренос $z_i^{'} = x_i + y_i + p_{i-1}$

Корекция	Разпознаващо правило
+1	Когато получената тетрадна сума на първия етаж е (0110), (0111), (1011), (1100), (1110) или (1111).
+1	Когато получената тетрадна сума на първия етаж е (0010) или (1001) и липсва тетраден пренос $p_3^{'} = 0$.
-1	Когато получената тетрадна сума на първия етаж е (0100)
-1	Когато получената тетрадна сума на първия етаж е (0010) или (1001) и има тетраден пренос $p_3^{'} = 1$.
+2	Когато получената тетрадна сума на първия етаж е (0011) или (0101) и липсва тетраден пренос $p_3^{'} = 0$.
+2	Когато получената тетрадна сума на първия етаж е (1010) и заедно с това ($x_2 = y_2 = 0$).
+2	Когато получената тетрадна сума на първия етаж е (0101) и заедно с това старшите битове на двете входни комбинации са еднакви, т.е. ($x_3 = y_3 = 0$) или ($x_3 = y_3 = 1$).

Във всички останали случаи корекция няма.

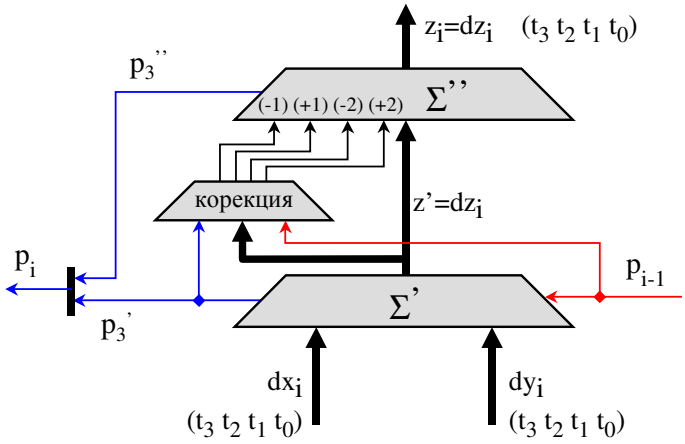
Пример

```
  1   1 1 1 1
  0 7 2 6 2 8 7
+ 0 8 1 6 7 1 9
  ---------------
  1 5 4 3 0 0 6
```

```
    1       1   1   1       1
 0000 1100 0011 1010 0011 1101 1100
+0000 1101 0001 1010 1100 0001 1111
 ---------------------------------
 0001 1001 0101 0101 0000 1111 1011
+     1111   10                1 1111
 ---------------------------------
 0001 1000 0111 0101 0000 0000 1010
```

От изложеното в този раздел, аналогично на случая според кодова таблица 2.1, се прави извод, че логическата структура на едноразрядния суматор в код 5211 за кодова таблица 2.2, ще съдържа два двоични 4-битови суматора и схема за генериране на корекциите. Тази логическа структура има вида, показан на фиг.5.1.

Логическата структура на суматора в съответствие с кодова таблица 2.2 реализира една корекция в повече (-2) в сравнение с тази, синтезирана в съответствие с таблица 2.1.



Фиг. 5.1. *Логическа структура на едноразряден BCD суматор в код 5211* (*таблица 2.2*)

6. Схема на допълнителен код за кодова таблица 2.2

Изложеното в раздел 4, относно изпълнението на операция изваждане, е напълно аналогично и в този случай. В сила са определения (4.1) и (4.2). В таблица 6.1 са изразени допълненията на десетичните цифри до 9, в съответствие с тяхната кодова таблица 2.2.

Таблица 6.1. Дефиниране на обратен код

Цифра d_k	Допълнение до 9	Код на цифрата $t_3t_2t_1t_0$	Инверсия на кода	Корекция	Код на допълнението
0	9	0000	1111		1111
1	8	0001	1110	-1	1101
2	7	0011	1100		1100
3	6	0101	1010		1010
4	5	0111	1000		1000
5	4	1000	0111		0111
6	3	1010	0101		0101
7	2	1100	0011		0011
8	1	1101	0010	-1	0001
9	0	1111	0000		0000

Както се вижда, кодът на допълнението може да се получи след побитова инверсия на кодовата комбинация на съответната десетична цифра и корекция (-1), ако е необходимо. Тъй като корекцията трябва да се постига автоматично, за целта се синтезира логическа функция за корекция. От таблица 6.1 като таблица на истинност следва представената карта на Карно (фиг.6.1) и логическото уравнение за функцията на корекция с (-1).

t3 t2 \ t1 t0	00	01	11	10
00	0	*	0	1
01	*	0	0	*
11	0	*	0	1
10	0	*	*	0

Фиг. 6.1. *Карта на Карно за функцията на корекция с* (-1)

$$K2.2_{(-1)} = (\overline{t_3} \cup t_2) \cap t_1 \cap \overline{t_0} \quad . \qquad (6.1)$$

Пример 6.1. Да се представи в допълнителен код числото X = -514816

$[X]_{ПК}$ = 1 1000 0001 0111 1101 0001 1010

Обратният код на това число се получава от правия след побитова инверсия и корекция в съответните тетради с (-1).

1	1000	0001	0111	1101	0001	1010	¬
1	0111	1110	1000	0010	1110	0101	+
0		1111		1111	1111		
1	0111	1101	1000	0001	1101	0101	

Допълнителният код се получава след прибавяне на единица към обратния код.

1	0111	1101	1000	0001	1101	0101	
						0001	+
1	0111	1101	1000	0001	1101	0110	
						1	+
1	0111	1101	1000	0001	1101	0111	

$[X]_{ДК}$ = 1 0111 1101 1000 0001 1101 0111

Пример 6.2. Да се изпълни операция събиране Z=X+Y, където X = -514816, Y = 287318. Резултатът трябва да бъде Z = -227498, получен в допълнителен код.

	1	1		1	1		
1	0111	1101	1000	0001	1101	0111	$= [X]_{ДК}$
							+
0	0011	1101	1100	0101	0001	1101	$= [Y]_{ДК}$
1	1011	1011	0100	0111	1111	0100	+
0	1	1111	1111	1	1	1111	
1	1100	1010	0011	1000	0000	0011	$= [Z]_{ДК}$

Проверка:

1	1100	1010	0011	1000	0000	0011	$= [Z]_{ДК}$
1	0011	0101	1100	0111	1111	1100	¬
0							+
1	0011	0101	1100	0111	1111	1100	
						0001	+
1	0011	0011	1100	0111	1111	1101	
0							+
1	0011	0011	1100	0111	1111	1101	$= [Z]_{ПК}$

Z = -227498

Схемата за преобразуване на числа със знак от прав код в допълнителен, както и използването й за изпълнение на операция изваждане, е аналогична на логическата структура, представена на фиг.4.2. Разликата в нея е в това, че дешифраторите, които формират тетрадните корекции, реализират логическа функция (6.1), синтезирана в съответствие с таблица 6.1.

6. Заключение

Кратко изложеното проучване показва, че двоичната аритметика отстъпва все повече място на десетичната. Това се дължи от една страна на възможностите на технологиите за реализация, а от друга страна на реалната необходимост. Десетичен хардуер вече се реализира както в традиционни микропроцесорни системи, така и в схеми с програмируема логика. Самата кухня е разнообразна, което означава, че в изминалите няколко години все още не е постигнато единодушие и в научно направление изследванията продължават.

Настоящото изследване също е показателно за разнообразните възможности за избор. Освен за традиционните BCD-кодове, все още не са известни технически решения за други кодове в смисъла на тук представените. Както е показано в [1], броят на 4-битовите тегловно значими двоично-десетични кодове е значителен 29059430400.

В първи раздел беше отбелязано, че кодът 5211 има своите достойнства, за да бъде избран. Но и самият той предлага достатъчно разнообразие, за да не се счита това изследване за окончателно. От възможните 64 кодови таблици за код 5211, тук са представени техническите решения само за две. Резултатите са достатъчни за да се направи избор в полза на решението, съответстващо на кодова таблица 2.1. Краткото изложение е показателно за обема на усилията, които ще следва да се положат, ако бъде решено да се изследва пълното множество от възможни кодови таблици. Това явно е обект на отделно и самостоятелно изследване.

Литература

[1]. Тянев, Д.С., *Организация на компютъра.* Том 1. ISBN: 978-954-20-0412-7, Технически Университет - Варна, 2008.

[2]. Тянев, Д.С., *Организация на компютъра. Упражнения*. ISBN: 954-20-0258-0, Технически Университет - Варна, 2007.

[3]. Ercegovac, M.D., Lang, T., *Digital Arithmetic*, Morgan Kaufmann Publishers, 2004.

[4]. Parhami, B., *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, ISBN 0-19-512583-5, 1999.

[5]. IBM Corporation, *General Decimal Arithmetic*, 2004.
Available at: http://www2.hursley.ibm.com/decimal .

[6]. IBM Corporation: *Decimal Arithmetic FAQ. Part 1 – General Questions* (2014).
Available at: http://speleotrove.com/decimal/decifaq1.html#emphasis .

[7]. Cowlishaw, M.F., *General Decimal Arithmetic Specification*. Version 1.70, IBM. 2009.
Available at: http://speleotrove.com/decimal/ .

[8]. Cowlishaw, M.F., *Decimal Floating-Point: Algorism for Computers*, Proceedings of the 16th IEEE Symposium on Computer Arithmetic, 2003.
Available at: http://www.cs.tufts.edu/~nr/cs257/archive/mike-cowlishaw/decimal-arith.pdf .

[9]. Duale, A.Y. et al, *Decimal floating-point in z9: An implementation and testing perspective*, IBM Journal of Research and Development , vol. 51, pp. 217-227, 2007.

[10]. Schwarz, E.M., J.S. Kapernik, M.F. Cowlishaw. *Decimal floating-point support on the IBM System z10 processor*. IBM Journal of Research and Development, Vol.53, Issue: 1, Jan.2009.

[11]. Chamkur, V.V. *Design and Implementation of IEEE-754 Addition and Subtraction for Floating Point Arithmetic Logic Unit*. International Journal of Scientific and Engineering Research. Vol. 3, Issue 7, July 2012. pp.1132-1138.

Available at: http://www.ijser.org/onlineResearchPaperViewer.aspx?Design-and-Implementation-of-IEEE-754-Addition-and-Subtraction-for-Floating-Point-Arithmetic-Logic-Unit.pdf .

[12]. Malte Baesler, Sven-Ole Voigt. *Analysis of Fast Radix-10 Digit Recurrence Algorithms for Fixed-Point and Floating-Point Dividers on FPGAs*. International Journal of Reconfigurable ComputingVolume 2013.
Available at: http://www.hindawi.com/journals/ijrc/2013/453173/ .

[13]. Number Systems.
Available at: http://dodiyahiten.files.wordpress.com/2012/09/chapter-2-number-systems.pdf .

[14] Rekha K James. *Design and synthesis of efficient mac architectures for high speed decimal processor*. PhD Thesis. Cochin University of Science and Technology. India. 2010. Available at: http://dyuthi.cusat.ac.in/xmlui/bitstream/handle/purl/3105/Dyuthi-T1079.pdf?sequence=1 .