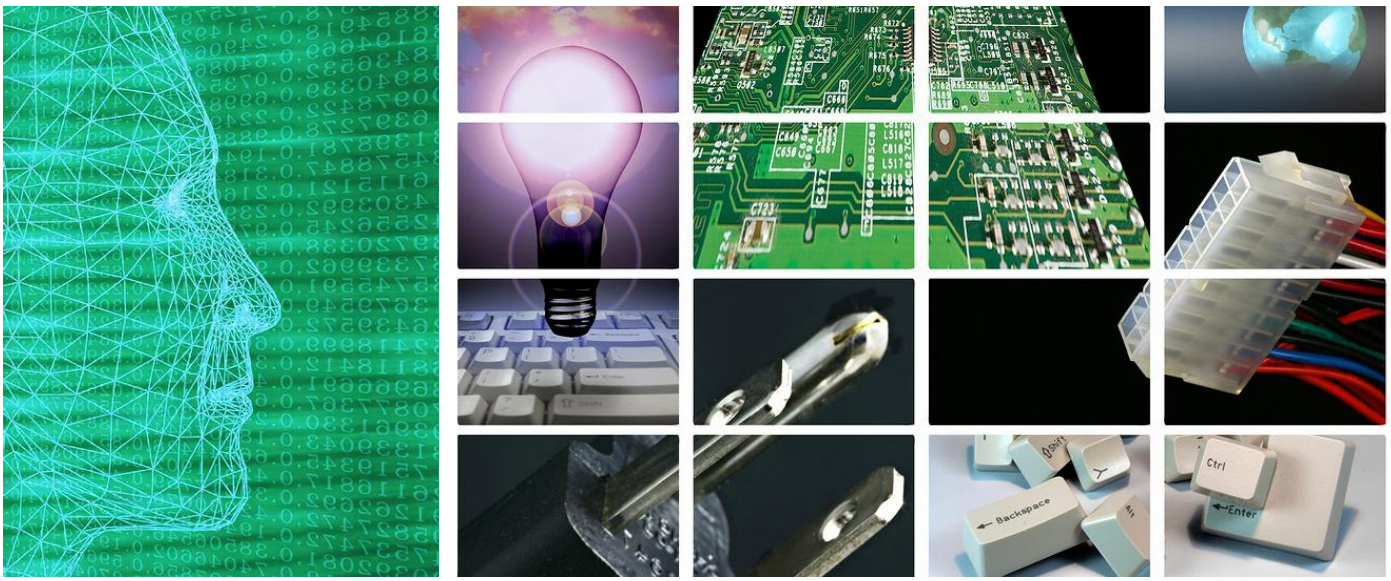
ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА
TECHNICAL UNIVERSITY OF VARNA

Година XII, Брой 3/2014

КОМПЮТЪРНИ НАУКИ И ТЕХНОЛОГИИ



IEEE Bulgaria Communications Chapter

FCA Faculty of Computing & Automation

COMPUTER SCIENCE AND TECHNOLOGIES

Варна, България

Year XII, Number 3/2014

Компютърни науки и технологии

Издание
на Факултета по изчислителна техника и автоматизация
Технически университет - Варна

Редактор: гл. ас. д-р Ю. Петкова
Гл. редактор: доц. д-р П. Антонов

Печат: ТУ-Варна

За контакти:

Технически университет - Варна
ФИТА
ул. „Студентска” 1, 9010 Варна,
България
тел./факс: (052) 383 320
e-mail: peter.antonov@ieee.org
yulka.petkova@tu-varna.bg

ISSN 1312-3335

Computer Science and Technologies

Publication
of Computing and Automation Faculty
Technical University of Varna

Editor: Ass. Prof. Y. Petkova, PhD
Chief Editor: Assoc. Prof. P. Antonov, PhD

Printing: TU-Varna

For contacts:

Technical University of Varna
Faculty of Computing and Automation
1, Studentska Str., 9010 Varna,
Bulgaria
Tel/Fax: (+359) 52 383 320
e-mail: peter.antonov@ieee.org
yulka.petkova@tu-varna.bg

ISSN 1312-3335

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 weighted code 5211

Dimitar S. Tyanev

Abstract: The structure synthesis of binary coded decimal adder in 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 all 64 possible 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

В случай че 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. ($x_3 \neq y_3$).

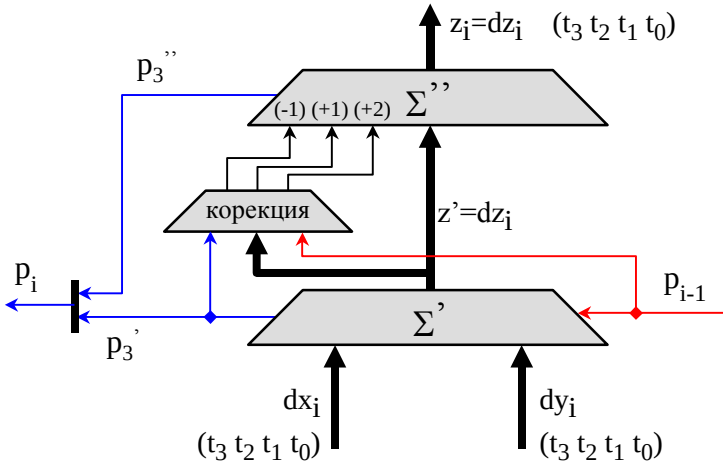
Във всички останали случаи корекция няма.

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 фигура 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

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

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.

$$[X]_{\text{ОК}} = \begin{cases} X, & X > 0; \\ 10^n - 1, & X < 0. \end{cases} \tag{2}$$

Определение (1) се постига поразрядно чрез разликата

$$(9 - d_k), \tag{3}$$

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 (фигура 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)

$$K2.1_{(-1)} = (t_2 \cap \overline{t_1}) \cup (t_1 \cap \overline{t_0}) \ . \qquad (4)$$

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.

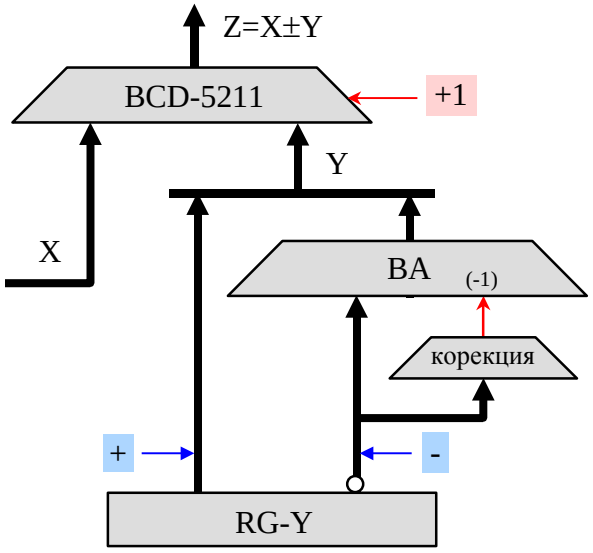
	1	1		1	1		
1	0111	0011	1000	0000	1111	0111	$= [X]_{ДК}$
							+
0	0011	1111	1011	0101	0001	1101	$= [Y]_{ДК}$
1	1011	0011	0011	0110	0001	0100	
							+
0				1		1111	
1	1011	0011	0011	0111	0001	0100	$= [Z]_{ДК}$

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 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 суматор BA (*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), 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). Конкретните правила 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 ($x_3 = y_3 = 1$).

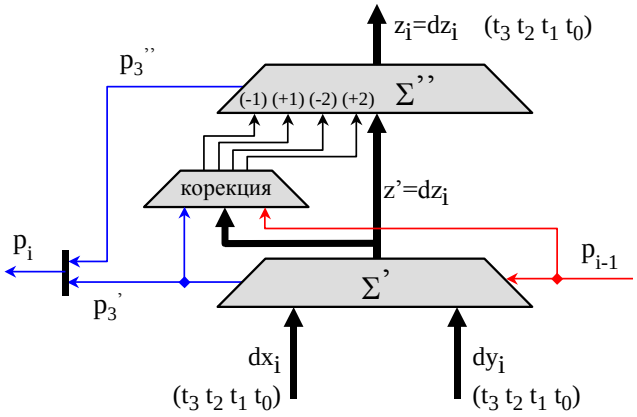
Във всички останали случаи корекция няма.

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 фигура 5.1.



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

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.

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 определенията (1) и (2). Така 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 (фигура 2) 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. 2. Карта на Карно за функцията на корекция с (-1)

$$K2.2_{(-1)} = (\overline{t_3} \cup t_2) \cap t_1 \cap \overline{t_0} \quad . \tag{5}$$

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

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

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 фигура 4.2. Разликата в нея е в това, че дешифраторите, които формират тетрадните корекции, реализират логическа функция (5), 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, 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.

За контакти:
Димитър С. Тянев,
www.tyanev.com

СЕРИЯ 4-ФАЗОВИ МИКРОКОНВЕЙЕРНИ АВТОМАТИ С ИЗПРЕВАРВАЩО НУЛИРАНЕ

Димитър С. Тянев

Резюме: Представен е анализ на 4-фазовия трансферен протокол. Направени са изводи и са формулирани технически изисквания към параметри, на които се основава проведения логически синтез. Представени са принципни логически схеми на серия 4-фазови микроконвейерни автомати и на свързани с тяхното функциониране логически схеми. Представено функционирането на логическите схеми. Изяснени са техните достойнства и недостатъци, което е предпоставка за тяхното приложение.
Ключови думи: микроконвейерен автомат, 4-фазов протокол, сигнал за фактическо закъснение.

Serial early-zero 4-phase micro-pipeline controllers

Dimitar S. Tyanev

Abstract: Analysis of the 4-phase transfer protocol is presented. Conclusions are made and are defined technical requirements to parameters that are base of the logical synthesis. Principal logical schemes of serial 4-phase pipeline controllers and additional circuits are presented. The functioning of these schemes is presented as well. Their advantages and disadvantages are defined which is a precondition fro their usage.
Key words: micro-pipeline, 4-phase protocol, signal completion detection.

1. Четири фазов конвейерен трансфер

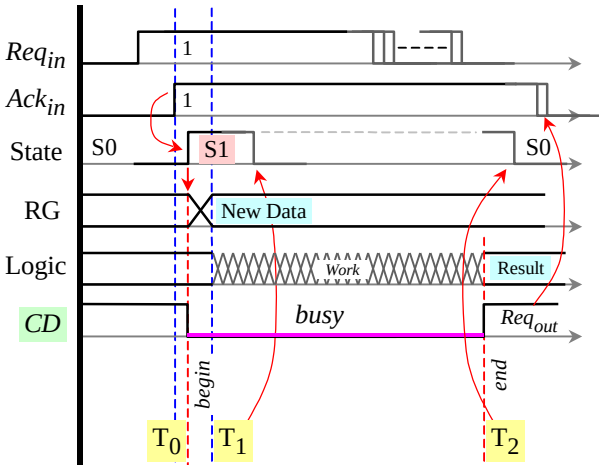
Управлението на асинхронни микроконвейери чрез 4-фазови конвейерни автомати се предопределя от използването на *Edge*-регистри фиксатори в структурата на микроконвейерните звена. От своя страна използването на 2-фазови конвейерни автомати налага фиксаторите да се реализират с *Latch*-тригери. Тъй като при тези автомати и двете състояния са работни, то *Latch*-тригерите трябва да имат структура DETFF (*Double-Edge Triggered Flip-Flop*), което се постига с нетрадиционни средства, увеличаващи значително апаратните разходи.

Управлявани от двойката входни сигнали (Req_{in}, Ack_{in}), 4-фазовите автомати се превключват в единично състояние винаги, когато и двата входни сигнала се установят в единично състояние. Предният фронт на новото състояние на автомата следва да се използва за осъществяване на данновия трансфер.

Със записа на нови данни в текущото звено, то стартира нови изчисления. За да се повтори този процес, конвейерният автомат на звеното трябва да възстанови изходното си състояние. Логично е това да стане в момент, когато микроконвейерното звено не е заплашено от това обратно превключване на автомата в изходно (нулево) състояние. За да бъде определен този момент, процесът на старт и функциониране на звеното е изследван чрез времедиаграмата от фиг. 1.

Както се вижда, след превключването на конвейерния автомат от състояние S0 в състояние S1, като следствие протичат две последователни превключвания. Първото превключване е на тригерите в регистъра фиксатор RG, който приема новите данни от предходното микроконвейерно звено. Второто превключване е на операционната логическа схема (*Logic*), която изчислява нов резултат. В началото на тези две закъснения (момент T0) сигналът на фактическото закъснение *CD* (*Completion detection*) пада в ниско ниво (логическа нула). Интервалът, в който този сигнал има ниско ниво, се приема за интервал, в който

микроконвейерното звено е в състояние “заето”, т.е. провеждащо текущото изчисление [3]. В момент T2 изчисленията завършват и сигналът *CD* се вдига във високо ниво (логическа единица). От този момент нататък резултатът е стабилен и може да бъде предаван към следващото звено. Единичната стойност на сигнал *CD* се възприема като сигнал за край на изчислението и може да се подава към следващия конвейерен автомат в качеството на сигнал заявка (Req_{out}).

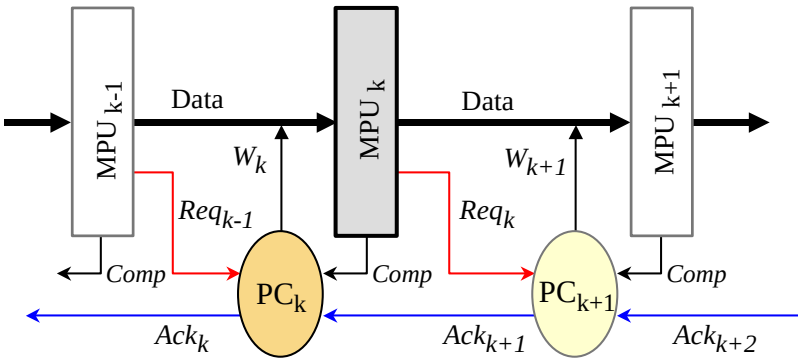


Фиг. 1. Времедиаграма за превключване на 4-фазов автомат

За да повтори трансферния цикъл, конвейерният автомат трябва да се превключи обратно в изходно състояния S0. Времевият интервал, когато възстановяването на автомата е правилно да стане, се определя от моментите T1 и T2. Минималното закъснение, с което възстановяването може да стане, се определя от времето за надеждно фиксиране на данните в конвейерния регистър. С други думи, латентността на тригерите в регистъра фиксатор определя минималната продължителност на единичното състояние S1. Следователно е необходим сигнал за потвърждение на края на записа в регистъра – сигнал *Complete*. Най-рано (момент T1) такъв сигнал може да генерира самия регистър. Най-късно сигналът *Complete* може да генерира операционната схема чрез сигнал *CD*, който се появява в момент T2. Всяко по-късно възстановяване на конвейерния автомат може да въведе непроизводително закъснение в превключването на автомат, а от там и в конвейера. Най-късното възстановяване може да предизвика сигналът потвърждение Ack_{in}, който ще се получи от следващия конвейерен автомат в отговор на предадените му данни. От казаното следва, че сигнал *Complete* има импулсен характер.

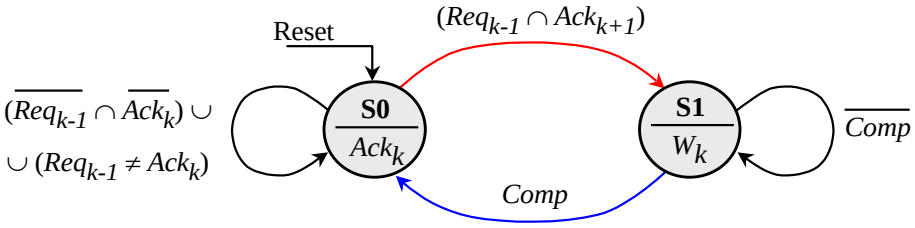
Фрагментът от фиг. 2 показва текущото (№k) микроконвейерно звено MPU_k (*Micro-pipeline Unit*) с конвейерния автомат PC_k (*Pipeline Controller*), който го управлява, както и връзките на тези два елемента със съседните в конвейера, в смисъла на казаното по-горе.

Появата на две единици за сигналите заявка Req_{k-1} и за потвърждение Ack_{k+1} причинява превключване на автомата в единично състояние и издаване на сигнал за запис W_k. Регистрите фиксатори на микроконвейерните звена записват входни данни само по един от фронтовете на импулсите W_k, $k = 0,1,2,3,\ldots$ За конвейерния автомат PC_k тези два сигнала са входни, т.е. $Req_{in} = Req_{k-1}$, $Ack_{in} = Ack_{k+1}$. Краят на записа на данните в регистъра фиксатор се отбелязва от сигнал *Complete* (*Comp*). Този сигнал се използва за превключване на конвейерния автомат обратно в нулево състояние. Приема се, че сигналът *Complete* има същата форма, както тази на сигнал *CD*, т.е. неговото ниско ниво трябва да се възприема като незавършил запис. Появата на преден фронт и високо ниво на този сигнал означава, че записът е завършил и тригерите на регистъра фиксатор са установени стабилно.



Фиг. 2. Структура на микроконвейер с 4-фазов автомат

Описаният протокол може да се изрази с граф на преходите, показан на фиг. 3.



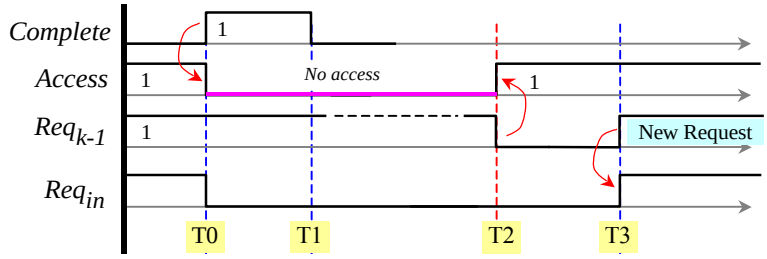
Фиг. 3. Граф на преходите на конвейерния автомат

В състояние S0 автоматът издава потвърждение Ack_k=1 за своето изходно състояние, а в състояние S1 потвърждението се снема, като в същото време се издава сигналът за запис W_k. Преход в състояние S1 се извършва при $Req_{k-1} = Ack_{k+1} = 1$. Обратният преход в S0 се осъществява с появата на сигнал *Comp*=1, след което сигналът за запис W_k пада в ниско ниво.

Връщайки се в състояние S0 (фиг. 3) съществува опасност от незабавно повторно и неправилно превключване на автомат PC_k обратно в състояние S1. На първо място като причина за това е импулсния характер на нулиращия сигнал. Като втора причина това е потенциалния характер на входните за автомата сигнали (Req_{in}, Ack_{in}), които след нулирането е възможно все още да продължават да са в състояние единица. Така съчетанието от тези две обстоятелства във времето обуславя възможността за ненадежност на състоянието S0. От описаното функциониране на 4-фазовите конвейерни автомати се разбира, че сигналите потвърждение падат в ниско ниво за кратко по време на данновия трансфер, т.е. през останалото време те са в активно високо ниво, затова са определени като потенциални. С други думи състоянието S1 е значително по-краткотрайно от изходното S0. Ето защо, след нулиране на такъв автомат, има голяма вероятност следващият микроконвейерен автомат PC_{k+1} (фиг. 2) бързо да възвърне потвърждението Ack_{k+1}=Ack_{in}=1. В същото време преходното звено (към което текущия автомат PC_k след възстановяването си в изходно състояние ще издаде потвърждението Ack_k) може да продължава да поддържа заявката си Req_{in}= Req_{k-1}. Вероятността за тази ситуация се дължи на това, че ако автомат PC_{k-1} е получил потвърждението Ack_k, но все още очаква сигнал заявка Req_{k-2}, той няма да може да се превключи. Това пък означава, че звеното, което той управлява, няма да може да снеме заявката Req_{k-1} и да я издаде повторно след новите изчисления. Така на входа на автомат PC_k ще продължават да стоят две единици Req_{k-1}=Req_{in}=Ack_{in}= Ack_{k+1}=1 и неговото лъжливо превключване в състояние S1 е напълно възможно.

Предотвратяването на изяснената по-горе опасност изисква функционирането на конвейерния автомат PC_k да бъде обвързано с поведението на заявката Req_{k-1}, която той получава от предходното микроконвейерно звено. Изводът, който може да се направи на

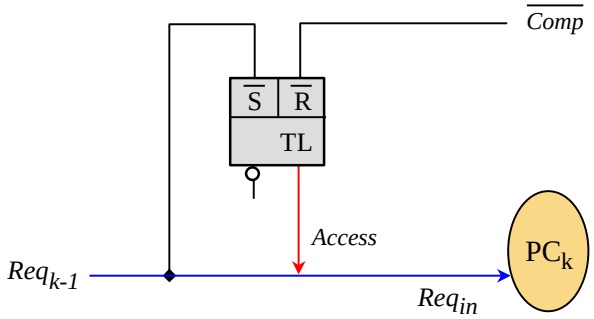
този етап, е че заедно с нулирането на конвейерния автомат, сигналът *Complete* трябва да блокира и действието на все още действащия сигнал Req_{k-1}. След изчезване на сигнал *Complete* автоматът не трябва да е в състояние да възприема заявката Req_{k-1}. Нещо повече, след изчезване на нулиращия сигнал, автоматът трябва да бъде в състояние да възприема тази заявка едва при нова (следваща) нейна поява. С други думи, с появата си сигнал *Complete* трябва да блокира достъпа на заявката Req_{k-1} до конвейерния автомат, а задния фронт на същата заявка, т.е. когато тя изчезне, да се отменя забраната и да разрешава следващия й достъп. Във времето това може да бъде изразено както е показано на фиг. 4.



Фиг. 4. Изключване и включване на достъпа на заявката от предходното звено

В момент T0 се появява сигнал *Complete*, който превключва конвейерния автомат в състояние S0. Достъпът на заявката Req_{k-1} до автомата е забранен и Req_{in}=0, но $Req_{k-1} \equiv Req_{in}$. В момент T1 изчезва сигнал *Complete*, но тъй като Req_{in}=0, автоматът не се превключва. В момент T2 достъпът се възобновява с изчезване на заявката Req_{k-1}. Към автомата се подава $Req_{k-1}=Req_{in}$=0. Няма превключване на автомата. В момент T3 се появява новата заявка Req_{k-1}=1, т.е. $Req_{k-1}=Req_{in}$=1 и от този момент нататък е възможно ново превключване на автомата в състояние S1.

Синтезирана логическа схема, осигуряваща достъп на заявката до конвейерния автомат според логиката на времедиаграмата от фиг. 4, е представена на фиг. 5.

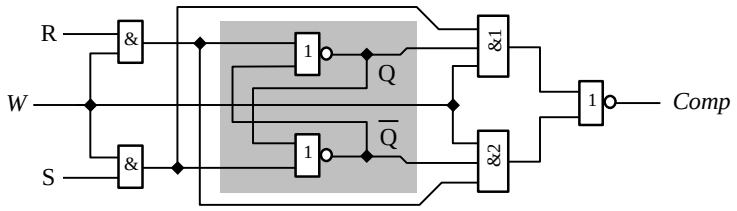


Фиг. 5. Логическа схема за генериране на сигнал Access

2. Сигнал *Complete*

2.1. Сигнал *Complete* за *Latch*-тригери

Ако се приеме, че сигнал *Complete* трябва да съобщи за края на записа в регистъра фиксатор, то това в частност е сигнал за край на превключването на един тригер. Микрооперацията запис е много кратка. Нейната продължителност е обикновено от 2.τ до 6.τ (с τ е означено времето за превключване на един логически елемент). На фиг. 6 е показана принципна логическа схема, която генерира сигнал *Complete*.



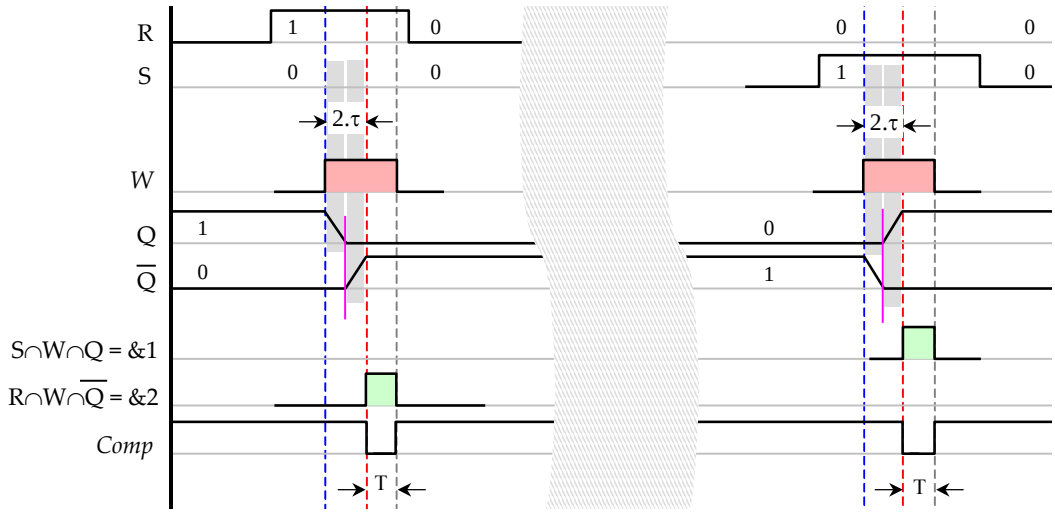
Фиг. 6. Сигнал Complete за край на запис в RS-тригер

Логическата функция, която реализира този сигнал се основава на функцията неравнозначност, която приема истинна логическа стойност след като аргументите на парафазната дизюнкция ($Q \cup \overline{Q}$) достигнат истинни логически стойности [1]. Парафазната дизюнкция е необходимо да бъде функционално потвърдена от логическите стойности по данновите входове. Освен това, тъй като при запис е възможно да не настъпи превключване на тригера, парафазната дизюнкция е функционално обвързана и със сигнала за запис *W*. В крайна сметка логиката на сигнала добива вида

$$Comp = \overline{(S \cap Q \cap W) \ \cup \ (R \cap \overline{Q} \cap W)} \qquad (1)$$

Въпреки, че схемата е приложена за елементарен RS *Latch*-тригер, тя може да се приложи и върху други по вид тригери. Схемата работи по един и същи начин и за двете превключвания 0→1 и обратно 1→0.

Функционирането на логическата схема е представено чрез времедиаграмата на фиг. 7.



Фиг. 7. Времедиаграма за край на превключването

Началното състояние на тригера е Q=1. С подаване на единица на R входа тригерът следва да се нулира Q=0, което се осъществява, когато се подаде сигнал за запис *W*. Както се вижда от времедиаграмата, за установяване на стабилното състояние са необходими минимум (2.τ) секунди. Това е собственото време за превключване на тригера от едно състояние в друго. Въпреки, че единицата на входа R продължава да е активна, други превключвания няма. На изхода *Comp*, който ни интересува, се вижда, че схемата генерира в отговор на превключването отрицателен импулс. Този импулс има активна стойност нула в интервала T. Той започва в момента на установяване на новото състояние на тригера и завършва в момента, в който изчезва сигналът за запис. С други думи, продължителността на сигнала *Comp* може да се изрази както следва

$$T_{comp} = T_w - 2.\tau \ , \qquad (2)$$

където с T_w е означена продължителността на сигнала за запис *W*.

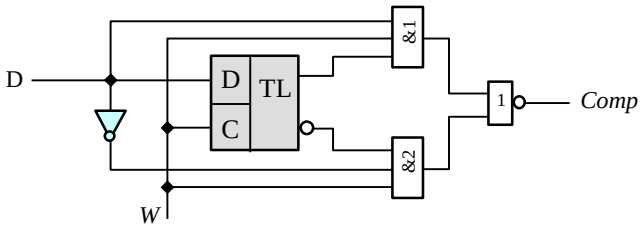
Аналогичен е процесът при обратното превключване, което причинява единицата постъпила по входа S. Задният фронт на сигнала *Comp* маркира момента, в който се завършва превключването на тригера.

Специално внимание заслужава случаят, при който тригерът не се превключва, тъй като записва същата стойност. В този случай сигналът *Comp* ще има формата на сигнала *W*. Тъй като схемата, чрез която се формира сигналът *Comp*, се изгражда само върху един тригер, превключването на ниво регистър става проблемно за регистриране, тъй като при запис някои от неговите тригери се превключват, но други могат да запазят състоянието си. По тази причина формирането на сигнала *Comp* само чрез един тригер не е пълноценно решение. Пълното решение изисква оборудване на всеки тригер със схема за формиране на сигнал $Comp_i$, $i=0,1,2,\ldots,(n-1)$. Чрез тези поразрядни сигнали следва да се формира конюнкцията

$$Comp = \bigcap_{i=0}^{n-1} Comp_i \quad . \tag{3}$$

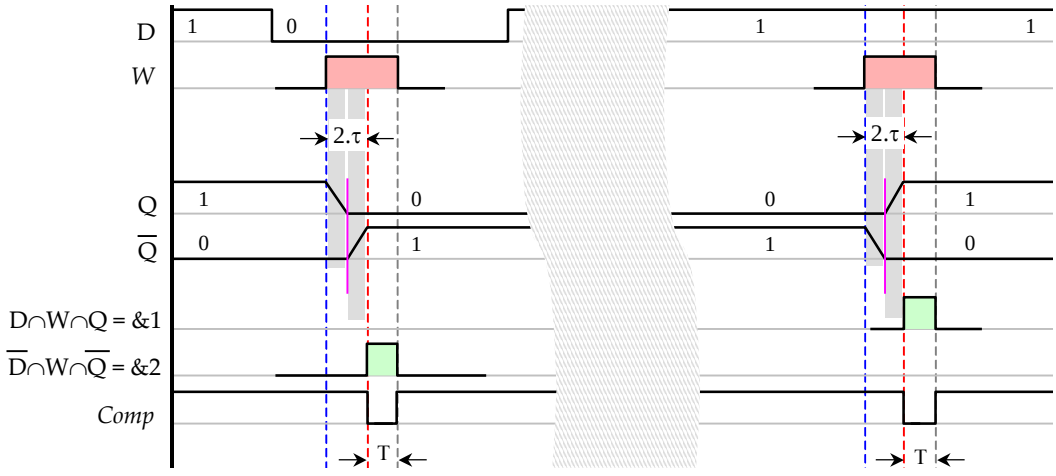
Така по-късите сигнали ще стробират по-продължителните, при което окончателният вид на сигнала *Comp* за целия регистър ще бъде като показания на фиг. 5. Въпреки това съществува вероятност да няма абсолютно никакви превключвания, което може да се случи ако новите данни съвпадат напълно със старите. В този случай конюнкцията (3) ще генерира сигнал, който отново ще съвпадне със сигнала за запис *W*, така както беше обяснен по-горе случаят с единичния тригер.

Тъй като регистрите фиксатори често се изграждат от D-тригери, на фиг. 8 е представена логическата схема на сигнал *Comp*, за този тип тригери.



Фиг. 8. Сигнал Complete за край на запис в D-тригер

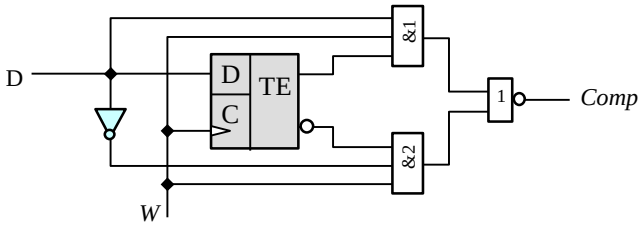
На фиг. 9 е представена времедиаграмата за превключване на схемата с D-тригер.



Фиг. 9. Времедиаграма за край на превключването

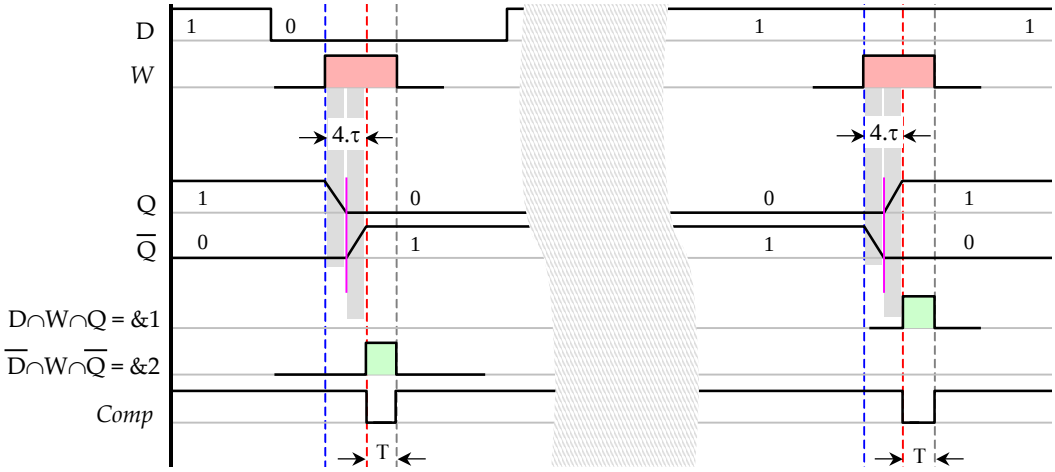
2.2. Сигнал *Complete* за тригери със структура *Edge* или *Master-Slave*, превключващи се по преден фронт

В раздел 1 бе отбелязано, че 4-фазовите микроконвейерни автомати са удобни за управление на данновия трансфер в случаи, когато входните регистри фиксатори на звената са изградени от противосъстезателни тригери. Такива са тригерите с *Edge*, или с *MS*-структура. По-долу е представена *Complete*-схема за *Edge* D-тригер, превключващ се по преден фронт. Съдържайки схемата на Хуфман, този тригер има време на превключване 4.τ [2]. В схемата от фиг. 10 той е представен без установъчните си входове. *Complete*-схемата е същата като тази от фиг. 8.



Фиг. 10. Сигнал Complete за край на запис в динамичен D-тригер

Единственото различие във времедиаграмата (фиг. 11) е по-голямото закъснение на сигнала *Comp* спрямо предния фронт на сигнала *W*.



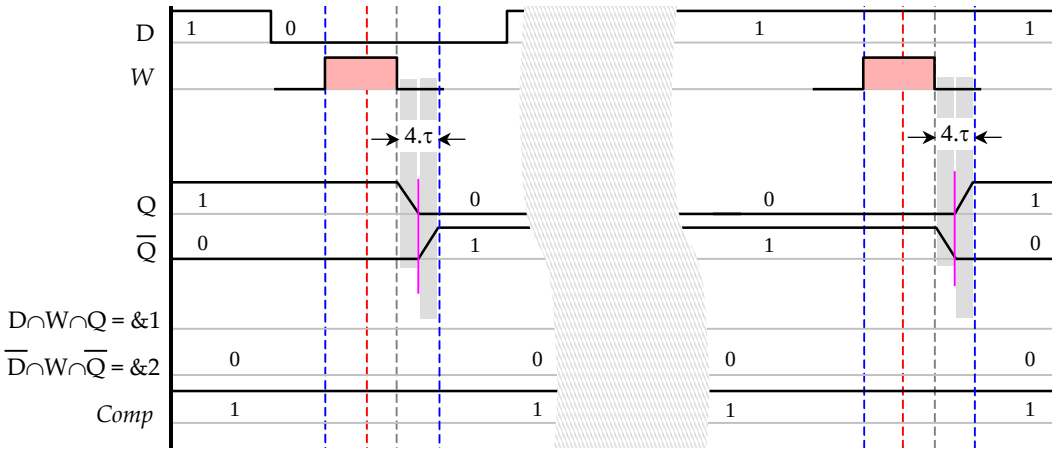
Фиг. 11. Времедиаграма за край на превключването

В случай на D-тригер със структура MS, логическата схема за сигнал *Comp*, както и времедиаграмата, остават същите като показаните на фиг. 10 и фиг. 11.

2.3. Сигнал *Complete* за тригери със структура *Edge* или *Master-Slave*, превключващи се по заден фронт

При тригери, които се превключват по заден фронт, функция (1) има константна стойност – логическа единица. Във времето процесът на запис има вида, показан на фиг. 12.

Трябва да се напомни, че първоначалната концепция за възстановяване на конвейерния автомат в състояние S0 предполага причината за това да бъде самият сигнал *Comp*. В настоящия частен случай обаче (фиг. 12), редът е обратен – трябва да се появи заден фронт в сигнала за запис *W*, след което се разпознава превключване. С други думи, изчезването на сигнала за запис, трябва да се породи от неговата поява. Тази последователност разбира се е абсурдна.



Фиг. 12. Времедиаграма за край на превключването

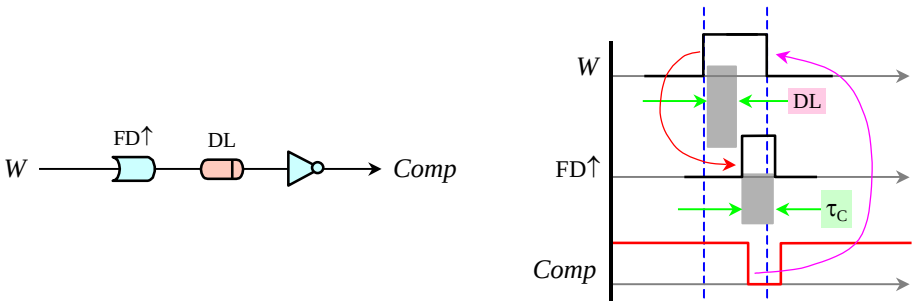
Невъзможността да се разпознае началото и края на превключването на тригерите при запис, изисква друга схема за формиране на сигнал *Comp*. Следва да се разбира, че използването на тригери, превключващи се по заден фронт, е неудачно решение още и по причина на непроизводителния времеви интервал на самия сигнал за запис *W*.

2.4. Обобщение

Изложеният в този раздел анализ беше необходим за изясняване етиологията на този вид асинхронни сигнали и възможността за тяхното използване. Съществените моменти могат да бъдат обобщени така:

1. При запис на нови данни в регистъра фиксатор на микроконвейерното звено не са гарантирани превключвания на тригерите;
2. Превключванията зависят от структурата на тригерите и имат различна продължителност;
3. Функциите (1) и (3) са неприложими за тригери, превключващи се по заден фронт;
4. При наличие на горните констатации, сигналът за запис *W* е единственото безусловно събитие, което не зависи от вида и от броя на тригерите в регистъра фиксатор. Това означава, че основният аргумент на функцията за край на превключването, трябва да бъде именно той.

Горните обобщения налагат извода, че най-удачния родител на сигнал *Comp* е предния фронт на сигнала за запис *W*. Ако тригерите на регистъра фиксатор се превключват по преден фронт, логическата схема, която ще го реализира, може да бъде (1), допълнена според (3). Тази реализация обаче е твърде скъпа. Ето защо, като окончателно решение за сигнал *Comp*, се налага задържан във времето импулс от детектор за преден фронт на сигнала за запис *W*, както е показано на фиг. 13.



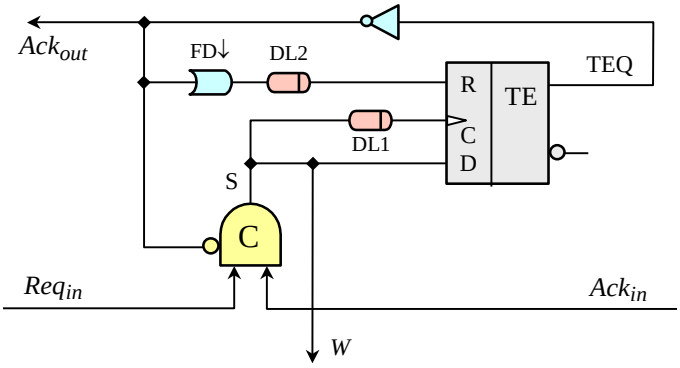
Фиг. 13. Схема за реализация на сигнал *Comp*

Техническите параметри на схемата трябва да удовлетворяват следните изисквания:

1. Продължителността τ_C на импулса FD↑ трябва да е по-голяма от времето за превключване на конвейерния автомата в състояние S0.
2. Закъснението DL на същия сигнал трябва да е по-голямо от времето за превключване на тригерите в регистъра фиксатор.

3. Микроконвейерен автомат с *Edge*-тригер

Решението за конвейерен автомат, представено на фиг. 14, който реализира 4-фазов протокол за трансфер, се основава на динамичен D-*Edge* тригер, работещ по преден фронт на сигнала за запис, подаван на вход C. Сигналът *Comp*, чрез който се възстановява изходното състояние на автомата, е генериран от схемата, представена на фиг. 13.

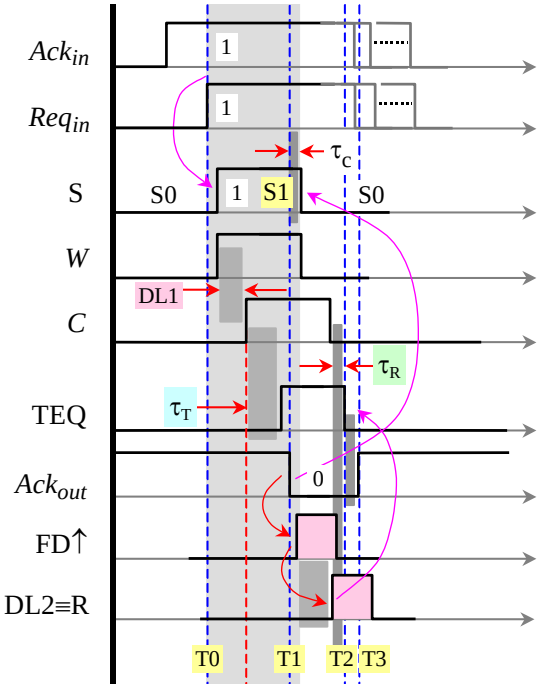


Фиг. 14. Четирифазов конвейерен автомат с динамичен тригер

Новото състояние S1 на C-елемента се подава на D входа на тригера, както и към регистъра фиксатор като сигнал за запис *W*. Новото състояние S1 се фиксира в тригера по предния фронт на сигнала, постъпващ на вход C. Времето за превключване на тригера е означено τ_T. Този сигнал е задържан във времето чрез симетричната *Delay*-верига DL1, което е необходимо за по-висока надеждност на записа. С инверсия правият изход на D-тригера се разпространява в обратна посока към предходния конвейерен автомат в качеството на сигнал потвърждение Ack_{out}. Същият сигнал се използва за нулиране на C-елемента. Възстановяване на автомата в изходно състояние S0 причинява сигнал *Complete*. Той идва в потвърждение на това, че микроконвейерното звено е приело надеждно в регистъра фиксатор новите данни. Схемата на този сигнал беше изяснена окончателно в раздел 2.4.

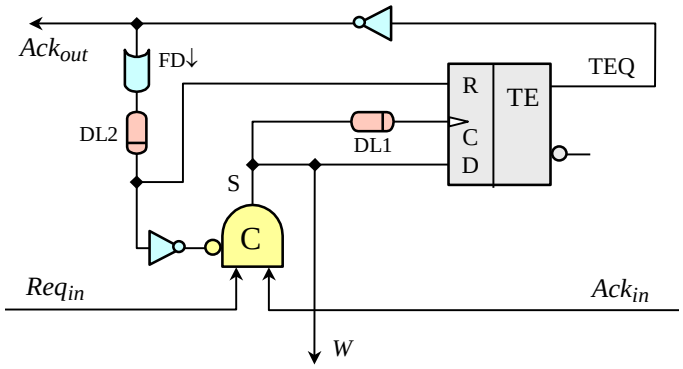
В схемата на конвейерния автомат с динамичен тригер, *Complete*-схемата е включена в обратна връзка към вход R за нулиране на тригера. Тъй като сигналът Ack_{out} е инверсен, е използван детектор на заден фронт FD↓. Може да се твърди, че сигналът Ack_{out} играе ролята на сигнал *Comp*. Така генерираният единичен импулс е задържан във времето чрез закъснителната верига DL2. Подаден на R входа той нулира тригера и C-елемента в схемата на автомата. Подробна времедиаграма за процеса на превключване на конвейерния автомат е представена на фиг. 15.

На времедиаграмата са отразени всички закъснения, които внасят последователно превключващите се логически елементи. Обратната връзка, която се образува от правия изход на D-тригера към нулиращия вход на C-елемента, не причинява състезания, което се осигурява от динамичния *Edge*-тригер. Продължителността на образувалият се сигнал за запис *W* е напълно достатъчна за надежден запис както за регистри с *Edge*-тригери, така и за регистри с MS-тригери. Времевият интервал, през който C-елемента се удържа принудително в нулево състояние се определя от ниското ниво на сигнала Ack_{out}. Към момента T3, в който принудителното удържане на C-елемента в нулево състояние се прекратява, сигналите Req_{in} и Ack_{in} са вече деактивирани.



Фиг. 15. Времедиаграма за превключване на автомата

Логическата схема от фиг. 14 има втори вариант, представен на фиг. 16.



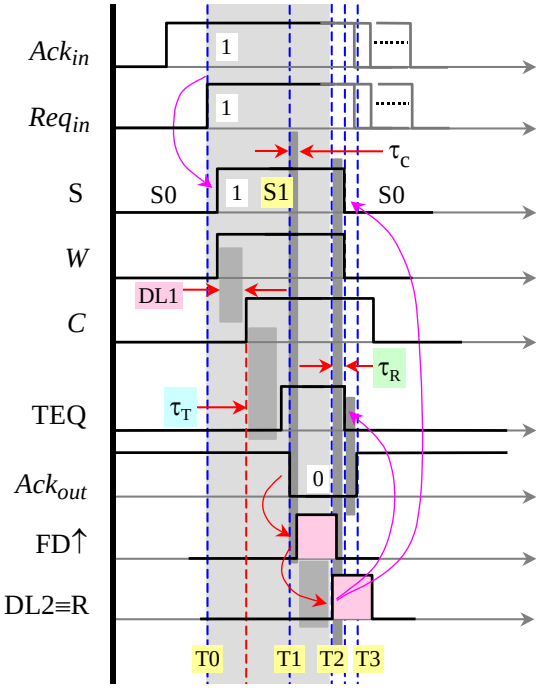
Фиг. 16. Четирифазов конвейерен автомат с динамичен тригер

Времедиаграмата на превключването на тази схема е малко по-различна в завършващата фаза, както е показано на фиг. 17. Различието се състои в удължаване на единичните фази на сигналите S и W.

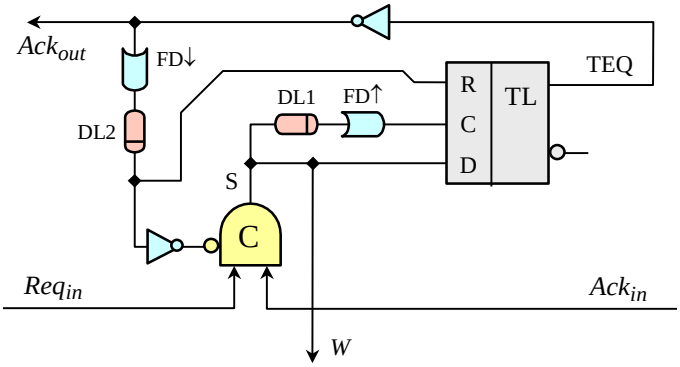
Времевият интервал, през който C-елемента се нулира и се удържа принудително в нулево състояние, е с начало момент T2 и край момент T3. Това дава допълнително време за деактивиране на сигналите Req_{in} и Ack_{in}. Именно това дава основание този вариант на схемата на автомата да се определи като по надеждна.

4. Микроконвейерен автомат с *Latch*-тригер

Логическата схема на конвейерния автомат може да бъде опростена чрез използване на обикновен (еднотактен) синхронен D-тригер със структура *Latch*. Тъй като *Latch*-тригерите работят по ниво, е необходимо за запис на новото състояние да се формира единичен импулс. За целта след закъснителния елемент DL1 е включен фронт детектор FD↑. Така в резултат на казаното се стига до логическа схема, представена на фиг. 18.



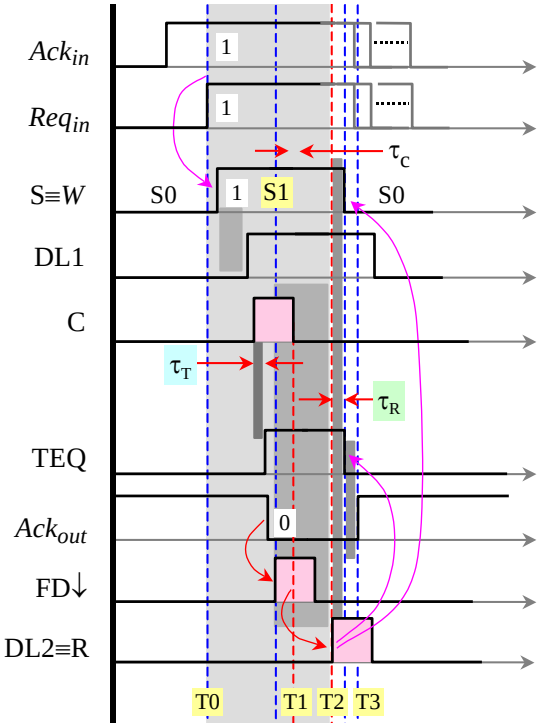
Фиг. 17. Времедиаграма за превключване на автомата



Фиг. 18. Четирифазов конвейерен автомат с Latch-тригер

На фиг. 19 е представена подробна времедиаграма на процеса на превключване на конвейерния автомат.

След превключване на входния C-елемент в единично състояние, към синхронния C-вход на тригера се подава със закъснение DL1 импулс, който записва единица. Чрез обратната връзка тази стойност се връща към предходния конвейерен автомат като сигнал за това, че конвейерният автомат е в процес на превключване, т.е. реализира данновия трансфер, който още не е завършил (Ack_{out}=0). Закъснението на този сигнал спрямо началото на сигнала за запис W в конвейерния фиксатор, което се сума от последователно превключващите се в тази верига логически елементи, е изразено на времедиаграмата. Възстановяването на автомата в изходно състояние S0 се реализира от задния фронт на появилия се сигнал Ack_{out}. Генерираният по този фронт импулс допълнително се задържа във времето чрез елемента DL2. Това закъснение трябва да гарантира, че пристигащият на входа R импулс за нулиране на тригера, няма да се застъпи във времето с по-рано появилият се импулс за запис на входа C. Закъснението (T2-T1)≠0 ще гарантира обратното превключване на тригера в нулево състояние. С нулирането на тригера и паралелно с него и на C-елемента, следват превключвания, които водят до изчезване на сигнала за запис W и до възстановяването на готовността на автомата (Ack_{out}=1).

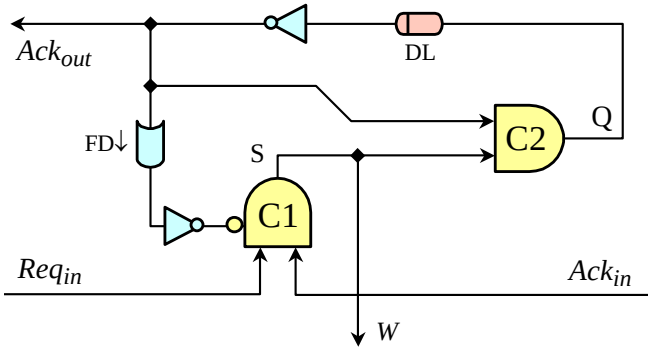


Фиг. 19. Времедиаграма за превключване на автомата

Логическата схема от фиг. 18 може да бъде реализирана и чрез синхронен *Latch* RS-тригер. Входът D ще бъде заменен от вход S. Останалото, казано по-горе, ще бъде в сила и за такава схема.

5. Микроконвейерен автомат с два С-елемента

Функционирането на синхронния D-*Latch* тригер в логическата схема от фиг. 18 наподобява това на С-елемент, ето защо ще бъде разгледан вариант на 4-фазов конвейерен автомат с два С-елемента. Вторият С-елемент трябва да се превключва в единично състояние при наличие на сигнал за запис *W* и сигнал от типа *Complete*. Генерирането на последния вече беше изяснено. Тук схемата за формиране на този сигнал е модифицирана. Използван е само един закъснителен елемент DL. Закъснението което той следва да осигури трябва да е достатъчно за надежден запис на данните в микроконвейерния регистър от сигнала *W*. Възстановяването на конвейерния автомат в изходно състояние S0 се осъществява чрез кратък нулев импулс, формиран по задния фронт на сигнала потвърждение Ack_{out}. Синтезираната логическа схема е представена на фиг. 20.

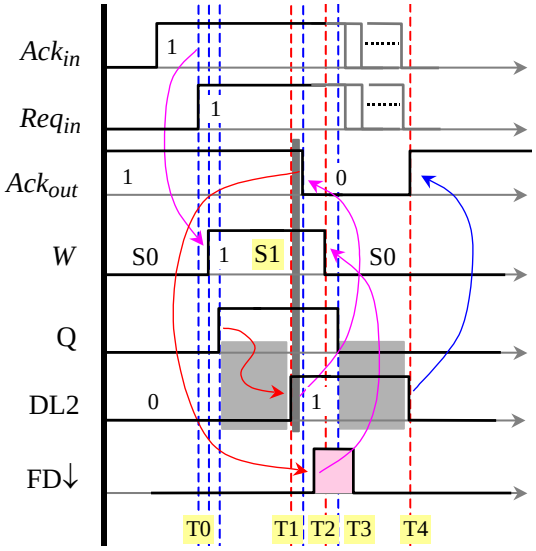


Фиг. 20. Принципна логическа схема на 4-фазов конвейерен автомат с два С-елемента

Времедиаграмата от фиг. 21 представя функционирането на конвейерния автомат. Изходното състояние на автомата е S0, в което той издава потвърждението Ack_{out}=1

(готовност за даннов трансфер). Превключването на автомата в състояние S1 започва в момент T0 при две входни единици (Req_{in} = Ack_{in} = 1). На времедиаграмата са изразени всички реални закъснения на превключващите се логически елементи.

Вторият C-елемент се превключва в единично състояние веднага с появата сигнала за запис *W*, тъй като в този момент Ack_{out}=1.



Фиг. 21. Времедиаграма за превключване на автомата

След закъснението, което внася елемент DL, в обратната връзка пропада сигнала за потвърждение на готовността, тъй като автоматът е в процес на трансфер. Освен това протича процес по възстановяване в изходно състояние. Той започва с кратка нула, подадена към първия C-елемент. Това води до прекратяване на импулса за запис W и веднага след това до нулиране и на втория C-елемент. Със закъснение DL се възстановява и сигнала потвърждение Ack_{out}=1.

6. Заключение

Изследването на микроконвейерния 4-фазов трансферен протокол с изпреварващо нулиране разкрива неговите особености, възможности и опасности. Във връзка с това е изследвана и възможността за синтез на сигнал за фактическото закъснение на микроопоерация запис в регистър фиксатор. Изявена е възможността за самосинхронизация на възстановяването на автомата, която е реализирана чрез симетрична закъснителна верига. Нейното закъснение е константно и при практическа реализация следва да се настройва със запас спрямо времето за превключване на избраните за регистъра фиксатор тригери. Функционирането на окончателната логическа схема на конвейерния автомат е подроно илюстрирано чрез времедиаграма.

Литература:

[1]. Тянев, Д.С., Синтез на асинхронни микроконвейерни системи с обща структура, Дисертация за Доктор на науките, Технически Университет - Варна, Ноември 2012.

[2]. Тянев, Д.С., Организация на компютъра. Проектиране на логически структури. Технически Университет - Варна, ISBN: 954-20-0259-9, 2004 год.

[3]. Тянев, Д.С., Асинхронно микроконвейерно звено с цифров компаратор, Компютърни науки и технологии, Технически Университет - Варна, ISSN 1312-3335, XI, том 1, 2013. стр. 18-22.

За контакти:
Димитър С. Тянев, www.tyanev.com

АППРОКСИМАЦИЯ В ЗАДАЧЕ УПРАВЛЕНИЯ ХАРАКТЕРИСТИКОЙ ЦИФРОВОГО ФИЛЬТРА ДЛЯ СПЕЦИАЛИЗИРОВАННОЙ КОМПЬЮТЕРНОЙ СИСТЕМЫ

В. С. Ситников, Т. П. Яценко, И. С. Петров, Т. В. Ситников

Резюме: определены зависимости, которые можно использовать для получения линейной характеристики управления АЧХ при аппроксимации характеристики. Показана возможность такого управления.

LINEARIZATION IN A DIGITAL FILTER CONTROL CHARACTERISTIC PROBLEM FOR A SPECIALIZED COMPUTER SYSTEM

V.S. Sytnikov, T.P. Yatsenko, I.S. Petrov, T.V. Sytnikov

Abstract: dependencies that could be used to obtain linear characteristic of frequency response control are defined. The possibility of such control is shown.

ВВЕДЕНИЕ

Специализированные компьютерные системы (СКС) активно внедряются в различные сферы жизнедеятельности человека. Они предназначены для сбора и обработки информации от датчиков, принятия решения и выработки управляющего воздействия на объект управления или соответствующие механизмы. Для повышения эффективности подобных систем необходимо управлять характеристиками перестраиваемых частотно-зависимых компонент. Например, в большинстве СКС имеются компоненты предварительной обработки и фильтрации входных сигналов, в состав которых наиболее часто входят цифровые фильтры. Для этой цели чаще всего рекомендуется использовать полиномиальные цифровые фильтры, отличительной особенностью которых является плоская амплитудно-частотная характеристика (АЧХ) в полосе пропускания. К их числу относятся известные фильтры Бесселя, Баттерворта и Чебышева второго рода [1-3].

За счет изменения коэффициентов числителя и знаменателя передаточной функции возможно как комплексное, так и раздельное управление характеристиками фильтра [4]. Однако в большинстве случаев для повышения их эффективности и плавного управления необходима линейная характеристика управления АЧХ устройства фильтрации. Обычно фильтры высокого порядка реализуются за счет соединения фильтров низкого порядка для обеспечения независимой перестройки характеристик.

ИЗЛОЖЕНИЕ

Анализ влияния коэффициентов передаточной функции цифрового фильтра на АЧХ проведен по передаточной функции первого порядка

$$H(z) = \frac{a_0 \pm a_1 z^{-1}}{1 + bz^{-1}}, \qquad (1)$$

где a_0, a_1, b – соответственно действительные коэффициенты числителя и знаменателя, $\pm$ – в числителе соответствуют ФНЧ (+) и ФВЧ (-).

Для фильтров первого порядка коэффициенты числителя в общем случае равны ($|a_0|=|a_1|$) и являются коэффициентом усиления $k=|a_0|=|a_1|$. Тогда передаточную функцию (1) для фильтра нижних частот (ФНЧ) можно записать в виде:

$$H(z)=k\frac{1+z^{-1}}{1+bz^{-1}} \tag{2}$$

Из (2) следует, что линейное управление АЧХ возможно коэффициентом усиления k за счет изменения коэффициентов числителя, что характерно для адаптивных фильтров. Однако, для уменьшения влияния шума на полезную составляющую сигнала при обработке необходима перестройка частоты среза фильтра.

В работах [4, 5] показаны возможные пути перестройки АЧХ ФНЧ и ФВЧ, а также трудности перестройки, вызванные тем, что коэффициенты числителя и знаменателя нелинейно зависят от частоты.

Для линеаризации характеристики управления АЧХ необходимо знание рабочего диапазона управления и заданной точности линеаризации. В большинстве случаев при высокой точности линеаризации диапазон управления разбивается на участки линейного управления, т.е. осуществляется кусочно-линейная аппроксимация. Полученная, таким образом, система линейных уравнений с ограничениями используется СКС для перестройки АЧХ цифрового фильтра.

Следует отметить, что при управлении цифровым фильтром необходимо осуществлять перестройку частоты среза с целью уменьшения влияния помех на полезную составляющую входного сигнала. В работе [5] получена зависимость нормированной частоты среза $\bar{\omega}_c$ ФНЧ в зависимости от коэффициента b знаменателя передаточной функции фильтра Чебышева второго рода и уровня пульсаций в полосе задержания RS (в dB), рис. 1

$$\bar{\omega}_c=\arccos\left(-\frac{1-2c^2\dfrac{1+b^2}{(1+b)^2}}{1-4c^2\dfrac{b}{(1+b)^2}}\right), \tag{3}$$

где $c^2=\dfrac{1}{\sqrt{10^{0.1RS}}}$, $\bar{\omega}_c=2\pi\dfrac{f}{f_d}$, $\bar{\omega}_c\in[0,\pi]$, f, f_d – соответственно текущая линейная частота и частота дискретизации.

В работе [5] для аппроксимации приведенных зависимостей разработан алгоритм, который основан на Чебышевской аппроксимации [6]. На основе анализа характера аппроксимируемой зависимости алгоритм выбирает направление аппроксимации либо с начала кривой, либо с ее конца.

Наиболее важным участком алгоритма аппроксимации считается блок, в котором происходит определение граничной точки между двумя участками аппроксимации. На каждой итерации происходит смещение конечной точки участка аппроксимации на одну позицию и заполнение матрицы размером N×N (где N-количество точек, на которое разбивается участок, аппроксимируемой кривой) строками, каждая из которых содержит массивы точек для первого участка. Затем осуществляется пересчет коэффициента знаменателя b и нормированной частоты среза $\bar{\omega}_c$, с целью получения расчетной погрешности и ее сравнения со значением заданной погрешности. Однако данный алгоритм обладает рядом недостатков: сложностью вычислений, большим объёмом занимаемой памяти, относительно длительным временем выполнения.

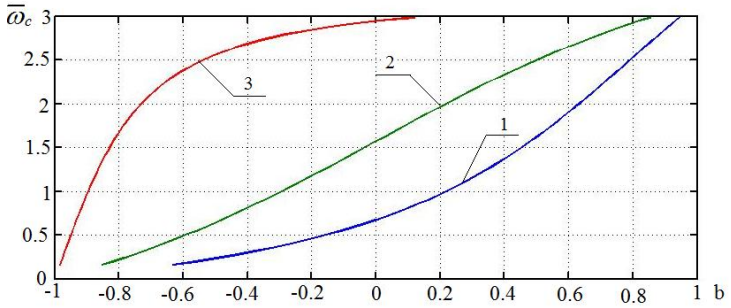


Рис. 1. График зависимости частоты среза фильтра $\overline{\omega}_c$ от коэффициента *b* знаменателя передаточной функции фильтра Чебышева второго рода при уровнях пульсаций в полосе задержания RS= 0,05 dB (1); 3 dB (2); 20 dB (3).

Поэтому возникла необходимость рассмотреть алгоритмы аппроксимации, которые позволят упростить алгоритм аппроксимации и объем вычисления, а чтобы ускорить процесс получения линейных участков управления в зависимости от заданной погрешности.

В соответствии с Чебышевской аппроксимацией кривая на участке аппроксимации должна находиться внутри “трубы”, рис. 2.

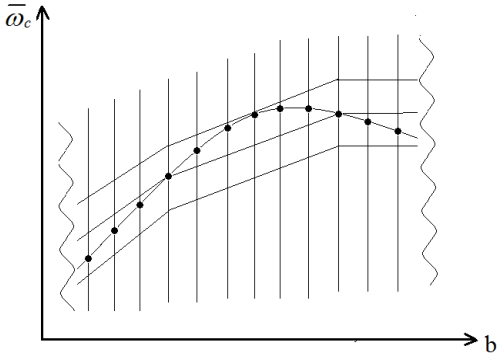


Рис. 2. Графическое пояснение к Чебышевской аппроксимации по методу “трубы”

В этом случае можно указать границы “трубы” на участке аппроксимации и сравнивать значения исходной кривой с границами “трубы”, заменяя вычисления значений сравнением текущего значения с границами “трубы”. Тогда возникает задача вычисления границ “трубы”. Для этого нет необходимости анализировать характер аппроксимируемой зависимости, а начиная с первой точки строить аппроксимирующую прямую к последующим точкам. Последовательность действий в этом случае может быть такой.

На первом этапе между первой точкой участка аппроксимации и n-ой точкой строиться прямая линия. На величину погрешности “трубы” определяется верхняя и нижняя ее границы.

Следует отметить, что может быть два пути определение границ. Первый путь – вычисления верхней и нижней границы по тем же крайним точкам участка. Второй – прибавления к найденной прямой величину погрешности “трубы”. Отметим, что относительная погрешность “вычислительного” пути от “найденного” может доходить до 1%.

На втором этапе осуществляется сравнение текущих значений с границами “трубы”. Если текущие значения находятся внутри “трубы” то берется следующее $n+1$-ое значение и процесс повторяется пока текущее значение кривой не превысит границы “трубы”. В этом случае процесс аппроксимации останавливается. Делается переход назад на одну точку (от $n+1$ к n) и осуществляется пересчет параметров аппроксимирующего участка.

Погрешность аппроксимации на участках показана на рис. 3, при заданной относительной погрешности $\delta = 0.01$. Алгоритм процесса аппроксимации по этому методу приведен на рис. 4.

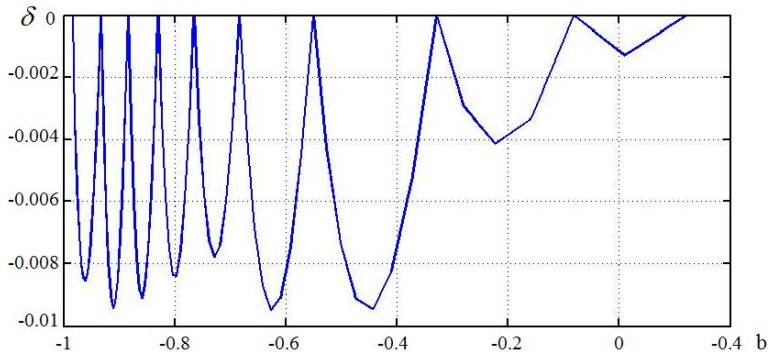


Рис. 3. Зависимость относительной погрешности аппроксимации δ по методу “трубы” от коэффициента b знаменателя передаточной функции фильтра Чебышева второго рода при уровне пульсации в полосе задержания RS= 20 dB

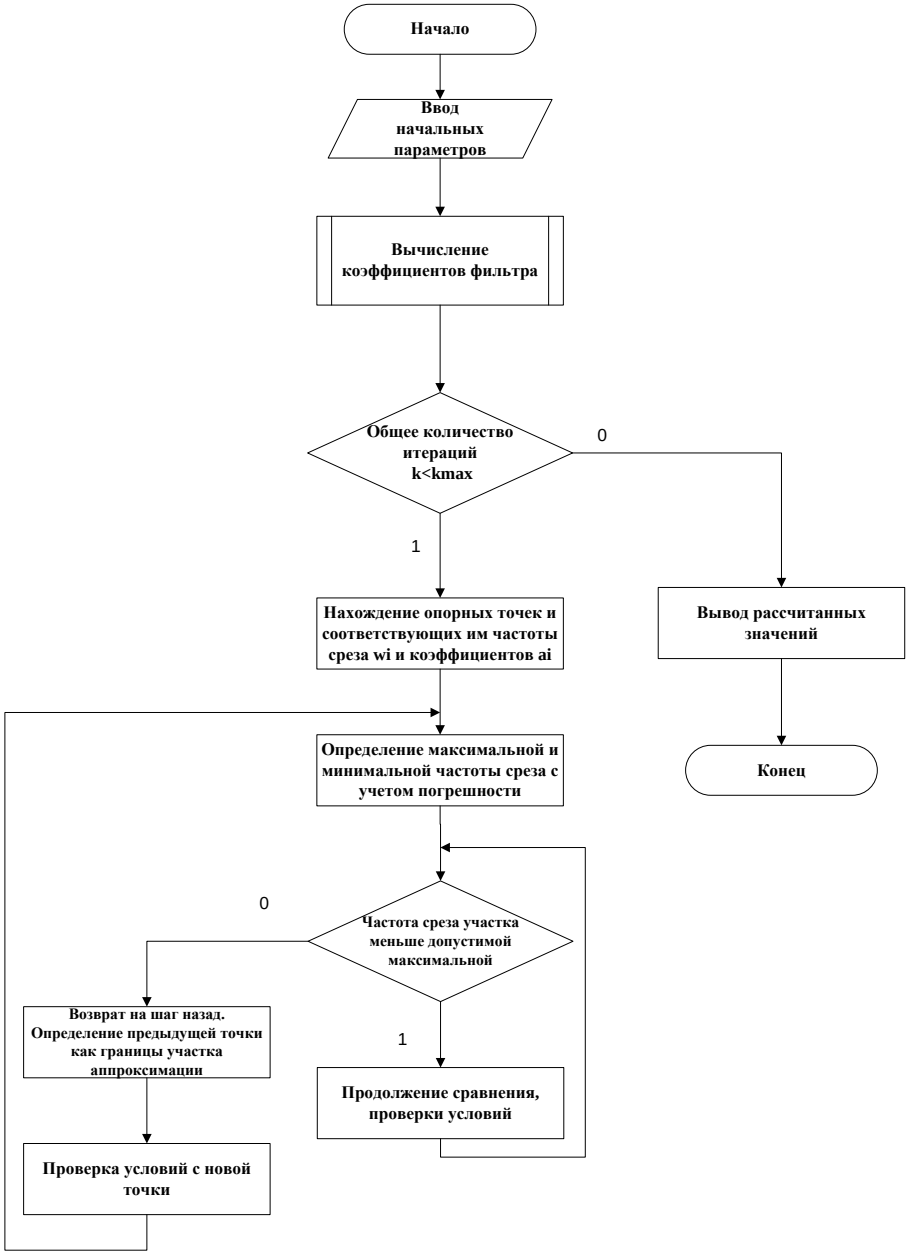


Рис. 4. Алгоритм аппроксимации зависимости частоты $\overline{\omega}_c$ среза фильтра от коэффициента b знаменателя передаточной функции фильтра Чебышева второго рода по методу “трубы”

В итоге аппроксимации получена система уравнений вида $y(k) = a_0 x(k) + b_0$,

$$\begin{cases} \bar{\omega}_c = 9.8820 \cdot b + 0.1571, & -0.9843 < b \le -0.9326, \\ \bar{\omega}_c = 8.8745 \cdot b + 0.6676, & -0.9326 < b \le -0.8840, \\ \bar{\omega}_c = 7.2745 \cdot b + 1.0996, & -0.8840 < b \le -0.8300, \\ \bar{\omega}_c = 5.4879 \cdot b + 1.4923, & -0.8300 < b \le -0.7656, \\ \bar{\omega}_c = 3.8254 \cdot b + 1.8457, & -0.7656 < b \le -0.6835, \\ \bar{\omega}_c = 2.3645 \cdot b + 2.1598, & -0.6835 < b \le -0.5506, \\ \bar{\omega}_c = 1.2387 \cdot b + 2.4740, & -0.5506 < b \le -0.3287, \\ \bar{\omega}_c = 0.6357 \cdot b + 2.7489, & -0.3287 < b \le -0.0816, \\ \bar{\omega}_c = 0.3864 \cdot b + 2.9060, & -0.0816 < b \le +0.1217, \end{cases} \quad (4)$$

В данном методе приходиться дополнительно вычислять границы трубы и проверять попадания в нее текущих значений заданной кривой. Однако можно пойти другим путем, когда по ходу анализа вычисляются значения заданной относительной погрешности аппроксимирующих значений прямой и текущих значений кривой (метод “веера”).

На первом этапе по этому методу также как и в предыдущем между первой точкой участка аппроксимации и n-ой точкой строиться прямая линия.

На втором этапе осуществляется вычисление относительной погрешности между значениями этой прямой и текущими значениями кривой. Если текущие значения имеют относительную погрешность меньше или равную заданной, то берется следующее $n+1$-ое значение и процесс повторяется пока для текущего значения кривой относительная погрешность будет больше заданной.

В этом случае процесс аппроксимации останавливается. Делается переход назад на одну точку (от *n+1* к *n*), осуществляется пересчет параметров аппроксимирующего участка. Погрешность аппроксимации на участках показана на рис. 5. Алгоритм процесса аппроксимации по методу “веера” показан на рис. 6.

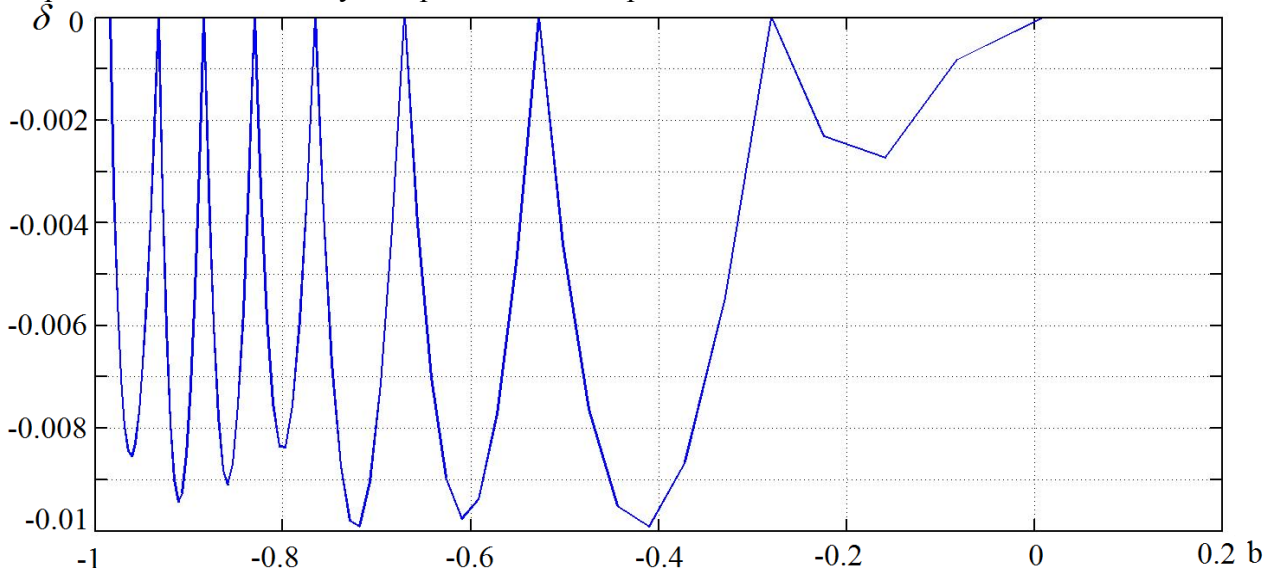


Рис. 5. Зависимость относительной погрешности аппроксимации δ по методу “веера” от коэффициента b знаменателя передаточной функции фильтра Чебышева второго рода при уровне пульсации в полосе задержания RS= 20 dB

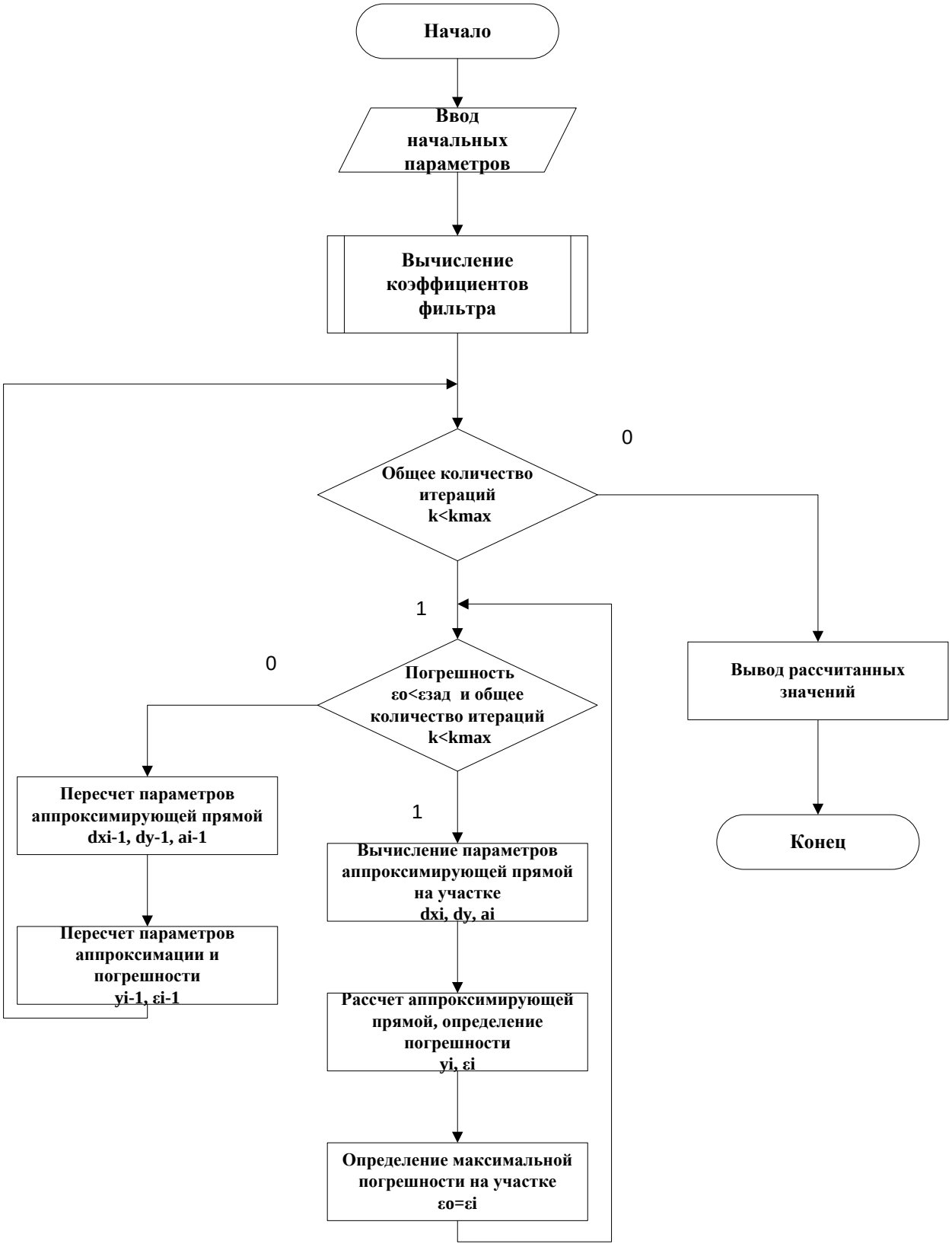


Рис. 6. Алгоритм аппроксимации зависимости частоты среза $\overline{\omega}_c$ фильтра от коэффициента *b* знаменателя передаточной функции фильтра Чебышева второго рода по методу "веера"

В итоге аппроксимации получена система уравнений аналогичная (1)

$$\begin{cases} \bar{\omega}_c = 9.8820 \cdot b + 0.1571, & -0.9843 < b \le -0.9326, \\ \bar{\omega}_c = 8.8745 \cdot b + 0.6676, & -0.9326 < b \le -0.8840, \\ \bar{\omega}_c = 7.2745 \cdot b + 1.0996, & -0.8840 < b \le -0.8300, \\ \bar{\omega}_c = 5.4879 \cdot b + 1.4923, & -0.8300 < b \le -0.7656, \\ \bar{\omega}_c = 3.7168 \cdot b + 1.8457, & -0.7656 < b \le -0.6705, \\ \bar{\omega}_c = 2.1972 \cdot b + 2.1991, & -0.6705 < b \le -0.5275, \\ \bar{\omega}_c = 1.1095 \cdot b + 2.5133, & -0.5275 < b \le -0.2798, \\ \bar{\omega}_c = 0.5826 \cdot b + 2.7882, & -0.2798 < b \le +0.0101, \\ \bar{\omega}_c = 0.3520 \cdot b + 2.9452, & +0.0101 < b \le +0.1217, \end{cases} \quad (5)$$

Сравнивая полученные результаты аппроксимации зависимости частоты среза фильтра $\bar{\omega}_c$ от коэффициента b знаменателя передаточной функции фильтра Чебышева второго рода при уровнях пульсаций в полосе задержания RS= 20 dB можно отметить, что с точностью до погрешности вычислений эти методы дают одинаковые результаты. Отличие начинается с пятой строки систем линейных уравнений (4) и (5), где сказываются вычислительные погрешности.

За счет оптимизации второго алгоритма можно упростить вычисления и ускорить процесс нахождения системы линейных уравнений при заданной погрешности аппроксимации δ.

На рис. 7 представлены АЧХ, построенные по формуле и аппроксимированным значениям коэффициентов *k* и *b* фильтра. Для уменьшения погрешности необходимо увеличить количество участков аппроксимации для достижения заданной точности воспроизведения АЧХ при ее перестройке.

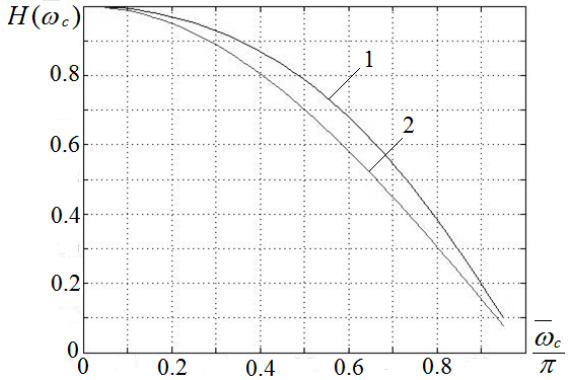


Рис. 7. Графики АЧХ построенные по формуле (1) и аппроксимированным (2) значениям коэффициентов *k* и *b* фильтра

ЗАКЛЮЧЕНИЕ

Таким образом, в результате аппроксимации получены системы линейных уравнений, описывающие характеристику управления, кроме того показана возможность такого управления.

ЛИТЕРАТУРНЫЕ ИСТОЧНИКИ

[1] Illiev G. Efficient design of adaptive complex narrowband IIR filters / Illiev G., Nikolova Z., Stoyanov G., Egiazarian K. / XII European Signal Processing Conference “EUSIPCO-2004” / 6-10 Sept., 2004, Vienna, Austria – p.p. 1597-1600.

[2] Устройство адаптивной фильтрации речевых сигналов “Золушка-микро-3“ http://speetech.by/manuals/manual_zolushka_micro3.pdf
[3] Сергиенко А.Б. Цифровая обработка сигналов / А.Б. Сергиенко — СПб.: Питер, 2006. — 751 с.
[4] Букашкин С.А. Справочник по расчету и проектированию ARC-схем; Под ред. А.А. Ланнэ — / С.А. Букашкин, В.П.Власов, Б.Ф.Змий и др. М. Радио и связь, 1984. — 368 с.
[5] Дикусар Е.В. Аппроксимация характеристики управления полиномиальной компонентой первого порядка / Е.В. Дикусар, А.А. Швец, Г.А.Грицкевич// Праці одеськ.політехн. ун-та – 2011 – Вип.1(35) – С. 141-146.
[6] Литовченко Н.М. Анализ критериев аппроксимации амплитудно-частотной характеристики устройства / Н.М. Литовченко, В.С. Ситников, А.В. Яковлев // Холодильна техніка і технологія. — 2006. — № 1(99). — С. 86—88.

Контакты:
д.т.н., проф. Валерий Ситников
кафедра “Компьютерные системы”
Одесский национальный
политехнический университет
E-mail: sitnvs@mail.ru

к.физ-мат.н., доц. Татьяна Яценко
кафедра “Высшей математики”
Одесский национальный
политехнический университет
E-mail: kuwtat@ukr.net

магистр Игорь Петров
кафедра “Компьютерные системы”
Одесский национальный
политехнический университет
E-mail: petrov.igor.od@gmail.com

бакалавр Тихон Ситников
кафедра “Компьютерные системы”
Одесский национальный
политехнический университет
E-mail: zzzvisperzzz@yandex.ru

ЛИНЕАРИЗАЦИЯ В ЗАДАЧЕ УПРАВЛЕНИЯ ХАРАКТЕРИСТИКОЙ ЦИФРОВОГО ФИЛЬТРА ДЛЯ СПЕЦИАЛИЗИРОВАННОЙ КОМПЬЮТЕРНОЙ СИСТЕМЫ

В.С. Ситников, Т.П. Яценко, И.С. Петров И.С., Т.В. Ситников

Резюме: определены зависимости, которые можно использовать для получения линейной характеристики управления АЧХ. Показана возможность такого управления.

LINEARIZATION IN A DIGITAL FILTER CONTROL CHARACTERISTIC PROBLEM FOR A SPECIALIZED COMPUTER SYSTEM

V.S. Sytnikov, T.P. Yatsenko, I.S. Petrov, T.V. Sytnikov

Abstract: dependencies that could be used to obtain linear characteristic of frequency response control are defined. The possibility of such control is shown.

ВВЕДЕНИЕ

В работе [1] рассмотрены вопросы аппроксимации характеристики управления цифрового фильтра, показаны достоинства и недостатки разных подходов аппроксимации характеристики управления.

Из анализа полученных результатов в работе [1] можно отметить, что при повышении точности (уменьшении погрешности) аппроксимации число участков аппроксимации резко возрастает. Это приводит к уменьшению этих участков и увеличению их количества, что затрудняет управление частотой среза в широких пределах, т.к. приходиться переходить от одного участка аппроксимации к другому. Кроме того, при малой погрешности управления и аппроксимации необходимо хранить большого количество значений коэффициентов участков, а также задавать значения коэффициентов с высокой точностью.

ИЗЛОЖЕНИЕ

В этом случае целесообразно ввести параметр управления, от которого частота среза фильтра изменялась линейно. Если частота среза $\bar{\omega}_c$ описывается уравнение

$$\bar{\omega}_c = \arccos\left(-\frac{1-2c^2\dfrac{1+b^2}{(1+b)^2}}{1-4c^2\dfrac{b}{(1+b)^2}} \right), \qquad (1)$$

где $c^2 = \dfrac{1}{\sqrt{10^{0.1RS}}}$, $\bar{\omega}_c = 2\pi\dfrac{f}{f_d}$, $\bar{\omega}_c \in [0,\pi]$, f, f_d – соответственно текущая линейная частота и частота дискретизации.

На рис.1 показаны графики зависимостей для цифрового фильтра первого порядка Чебышева второго рода.

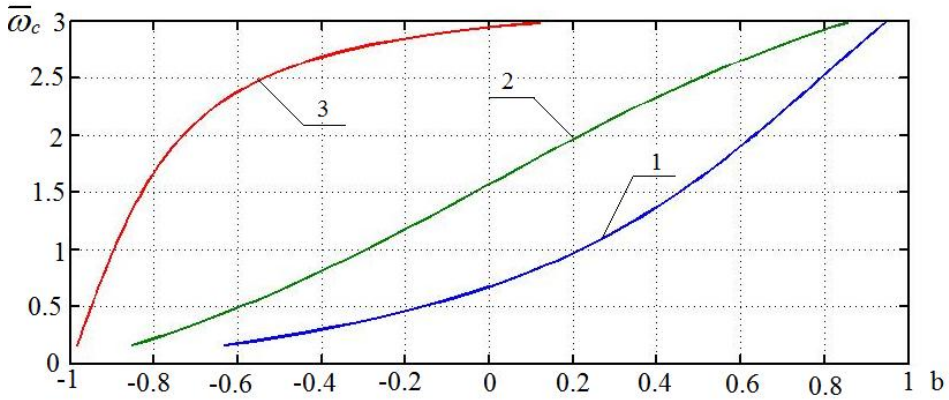


Рис. 1. График зависимости частоты среза фильтра $\bar{\omega}_c$ от коэффициента *b* знаменателя передаточной функции фильтра Чебышева второго рода при уровнях пульсаций в полосе задержания RS= 0,05 dB (1); 3 dB (2); 20 dB (3)

То введем параметр управления d и коэффициенты коррекции характеристики α и β так, чтобы $\bar{\omega}_c = \alpha d + \beta$. В этом случае

$$\cos(\alpha d + \beta) = -\frac{1 - 2c^2 \dfrac{1+b^2}{(1+b)^2}}{1 - 4c^2 \dfrac{b}{(1+b)^2}}.$$

Определим коэффициент b решая квадратное уравнение и учитывая условие устойчивости $|b| < 1$ получим:

$$b = \frac{-\left[\left(2c^2 - 1\right)\cos\left(\alpha d + \beta\right) - 1\right] - 2c\sqrt{1-c^2}\sin(\alpha d + \beta)}{\left(2c^2 - 1\right) - \cos\left(\alpha d + \beta\right)}. \qquad (2)$$

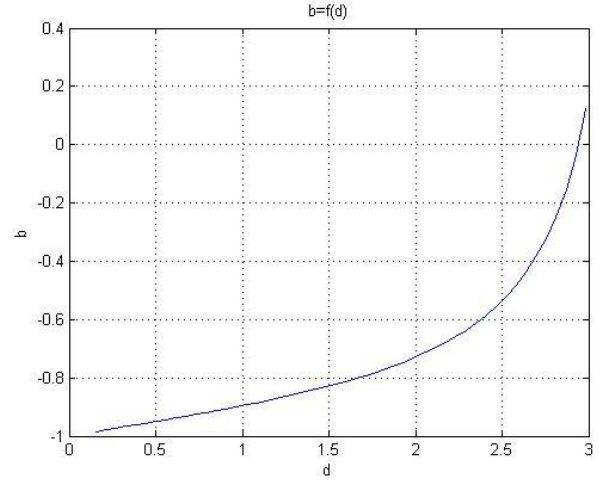
При реализации цифрового фильтра обычно его основные свойства не меняются, поэтому уровни пульсаций в полосе пропускания и задержания постоянны. Тогда в формулах (1) и (2) c=const, при этом 0<c<1. Для упрощения введем такой фиктивный угол ξ, чтобы $c = \cos\left(\frac{\xi}{2}\right)$.

В этом случае после преобразований формулу (2) можно представить в виде:

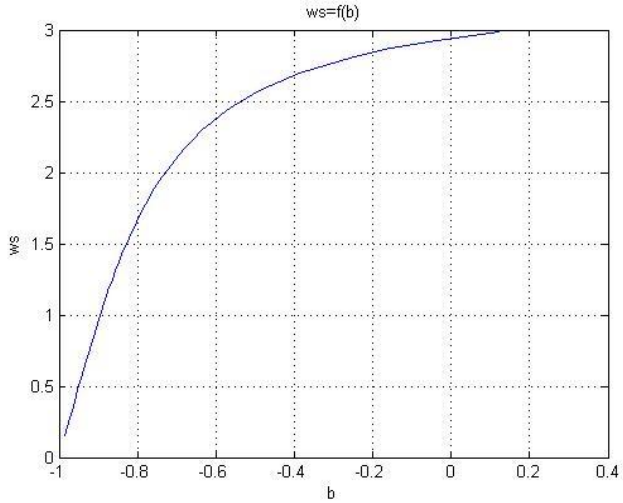
$$b = -\frac{\sin\left(\dfrac{\xi - (\alpha d + \beta)}{2}\right)}{\sin\left(\dfrac{\xi + (\alpha d + \beta)}{2}\right)}. \qquad (3)$$

Например, для случая показанного на рис 2:

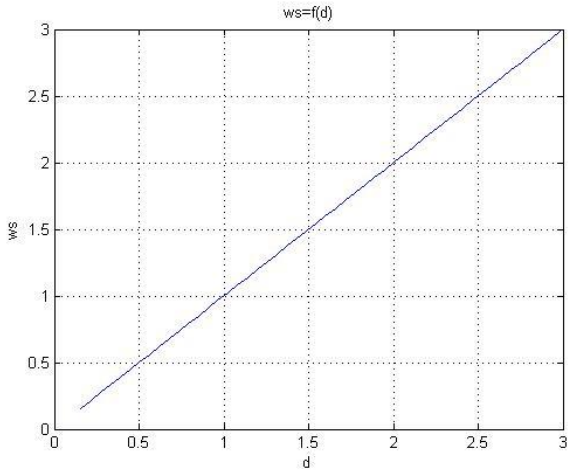
$$RS = 20\,dB;\ c = 0.1;\ \xi = 2.9413\,rad$$



а)



б)



в)

Рис. 2. График зависимостей коэффициента знаменателя *b* от параметра управления *d* (а), частоты среза ξ от коэффициента знаменателя *b* (б), а также частоты среза ξ от параметра управления *d* (в) для ФНЧ первого порядка Чебышева второго рода при *RS=20dB*

При этом график зависимости фиктивного угла ξ от уровня колебательности C имеет вид, приведенный на рис 3.

Для реализации на микропроцессорной технике соотношение (7) более приемлемо, т.к. значения функции могут быть заранее записаны в память. В этом случае обычным считыванием по аргументу функции можно получить значение функции синус.

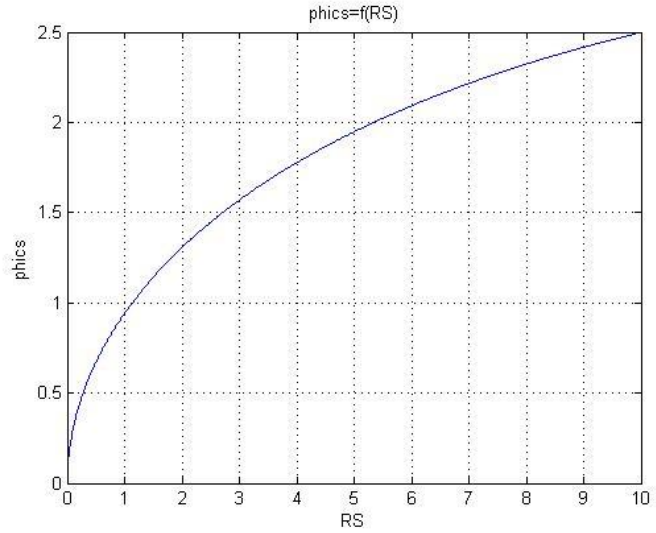


Рис. 3. График зависимости фиктивного угла ξ от уровня c

ЗАКЛЮЧЕНИЕ

Таким образом, в результате введения параметра управления d и линеаризации зависимости получаем линейную характеристику управления. При изменении параметра управления d осуществляется перестройка, как коэффициента усиления (2), так и частоты среза (4). Однако для изменения частоты среза при неизменной амплитуде необходима коррекция значения коэффициента усиления k при новом значении коэффициента знаменателя b.

В работе определены зависимости, которые можно использовать для получения линейной характеристики управления АЧХ, кроме того показана возможность такого управления.

ЛИТЕРАТУРНЫЕ ИСТОЧНИКИ

[1]. Ситников В.С., Аппроксимация в задаче управления характеристикой цифрового фильтра для специализированной компьютерной системы / В.С. Ситников, Т.П. Яценко, И.С. Петров, Т.В. Ситников, 2013

Контакты:
д.т.н., проф. Валерий Ситников
кафедра “Компьютерные системы”
Одесский национальный политехнический университет
E-mail: sitnvs@mail.ru

к.физ-мат.н., доц. Татьяна Яценко
кафедра “Высшей математики”
Одесский национальный политехнический университет
E-mail: kuwtat@ukr.net

магистр Игорь Петров
кафедра “Компьютерные системы”
Одесский национальный политехнический университет
E-mail: petrov.igor.od@gmail.com

бакалавр Тихон Ситников
кафедра “Компьютерные системы”
Одесский национальный политехнический университет
E-mail: zzzvisperzzz@yandex.ru

APPLICATION OF THE EXPERT SYSTEM BASED ON FUZZY LOGIC TO HEI'S AUTHORITIES TO CREATE MORE EFFECTIVE CURRICULA

Aleksandra Mreła, Oleksandr Sokołov

Abstract: Preparing good HEI's curricula is very difficult because there must be a lot of factors taken into account. Thus even if the curriculum is good, it must be checked and improved the next academic year. This paper presents a recurrence model of improving a curriculum for new students according to the results of current students based on fuzzy relations between assumed competences, subjects of a curriculum and acquirement of these competences by graduates and a methods for comparing graduates' competences with employers needs.
Keywords: quality of education, curriculum, fuzzy logic, max-min composition, optimization method

1. Introduction

The curricula of higher educational institutions are designed by the faculty, mostly in response to the needs of the job market. When the needs of the local job market are known, the process of creating a new curriculum begins. At the beginning of this process there are designed competences that prospective students should be taught and prospective graduates should acquire. After that, the designers have to choose how much hours of classes are devoted to help students acquire each of these competences and to construct the matrix – the correspondence between competences and subjects.

This matrix shows which subjects are devoted for teaching each competence. The assumed competences can be taught on one or on a few subjects. For example, let the competence c1, taught on the IT curriculum, be described as follows "Student can discuss the main ideas of IT in English". This competence can be taught on one subject called e.g. "English in computer sciences" or on a few subjects.

When, on one subject there are taught a few competences, the teacher has to distribute the time of the subject in such a way that he could help students achieve each competence in the sufficient way (Ref. 1). This distribution cannot be prepared once and for all because it depends on the students, their knowledge and skills, so in this situation the fuzzy logic can be really useful because phrases like "fluently", "rather fluently", "well", "poorly" and so on, are used.

The designers of the curriculum during building of the relation $R_{pc} : P \to C$ between subjects P and competences C, can make this relation be fuzzy by describing how much the competence is taught on the subjects using, for example, phases: "completely", "almost completely", "a little" and so on or they can put numbers from the interval [0,1] which show the degree of acquirement of competence on the subject. For example, 1 means students can acquire the competence totally on this subject and 0.1 means students can acquire the competence a little on this subject.

2. The solution of the problem

Let us define four sets: $S = \{s_1, \ldots, s_K\}$ – the set of students; $P = \{p_1, \ldots, p_N\}$ – the set of subjects; $C = \{c_1, \ldots, c_M\}$ – the set of competences; $T = \{t_1, \ldots, t_N\}$ – the percentage of time of the curriculum which is devoted to the subject from the set P.

Let us put the coefficients of the relation Rpc into Table 1.

Table 1. The relation R_{pc} between subjects S and competences C which are taught on the subjects

Competences / Subjects	c_1	…	c_M	Time for the subject
p_1	pc_{11}	…	pc_{1M}	t_1
…		…		…
p_N	pc_{N1}	…	pc_{NM}	t_N

The coefficients pc_{ij} (i=1, …, N; j=1, …, M) of the relation R_{pc} denote the strength (ability) of the subject p_i to convey the competence c_j which shows how important, according to the designers of the curriculum, the subject p_i is for teaching the competence c_j. Because of that we can assume that, the coefficients p_{ij} (i=1, …, N; j=1, …, M) belong to the interval [0,1] and $\sum_{j=1}^{M} pc_{ij} = 1$.

Let us build the second relation $R_{sp} : S \to P$ between subjects and students, where sp_{ik} (i=1,…,N; k=1,…,K) denote students' marks for their subjects assessments measured as the percentage of the highest grade (sp_{ik}=1 means that the student k has completed the subject i very well and sp_{ik}= 0 means that the student k has not received credit for the subject i).

Table 2. The relation R_{sp} between students and subjects

Subjects / Students	p_1	…	p_N
s_1	sp_{11}	…	sp_{1N}
…		…	
s_K	sp_{K1}	…	sp_{KN}

On the basis of these relations R_{pc} and R_{sp}, the new relation $R_{sp} \circ R_{pc} = R_{sc} : S \to C$ can be built (table 3). This relation describes the level of acquirement of the competences by the students measured by their results of the assignments.

Table 3. The relation R_{sc} between students S and competences C which they are taught

Competences / Students	c_1	…	c_M
s_1	sc_{11}	…	sc_{1M}
…		…	
s_K	sc_{K1}	…	sc_{KM}

Using the relation R_{sc} there can be found the subset of students who have acquired the competences on the desired levels $C^* = \{c^*_1, c^*_2, ..., c^*_M\}$ (this is the inverse task which is solved using inverse relationships). For instance, such tasks arise when business people have been looking for talented students. We name this task as ***task 1.***

On the basis of the relation R_{sc}, the average level of acquirement of the competences after assignments can be estimated and then the heads of HEIs can change the curriculum, that means they can *correct the time devoted for subjects* (Fig.1) for getting the *desired average levels of competences*. Let this task be ***task 2***.

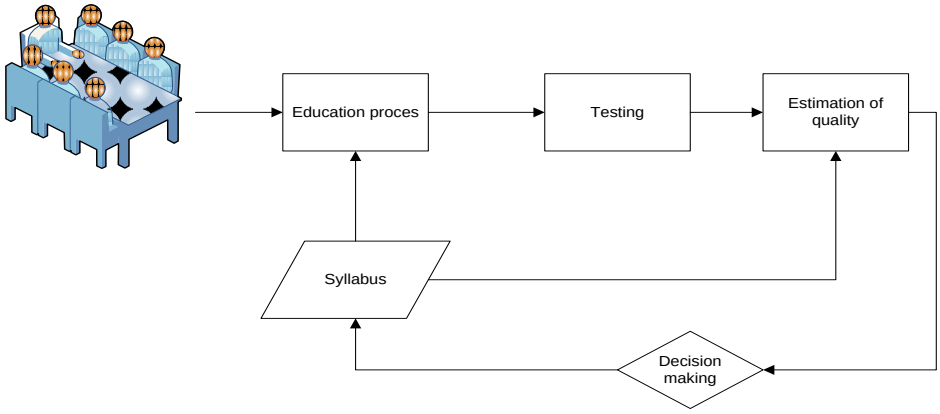


Fig. 1. The control system of the education process

Notice that the process shown in Fig. 1 will never stop because every year HEIs accept new groups of students and the curricula have to be improved according to constant changes in the law and new scientific discoveries.

Let the curriculum consist of N subjects, and the distribution of curriculum time be given by the set $T = \{t_1, t_2, \ldots, t_N\}$. Of course $\sum_{i=1}^{N} t_i = 1$.

We can calculate the time $T_C = \{tc_1, ..., tc_M\}$ that is taken for teaching each competence c_j during the study according to the curriculum as follows:

$$tc_j = \sum_{i=1}^{N} pc_{ij} \cdot t_i,\ j = 1, ..., M,\ pc_{ij} \in [0,1]. \tag{1}$$

Based on the relation $R_{sc} : S \to C$ obtained after taking examinations there can be calculated the vector C^{aver} of the normalized average competence values c_j^{aver} for each competence c_j (comp. table 4) as follows:

$$c_j^{aver} = \frac{\sum_{i=1}^{L} sc_{ij}}{\sum_{i=1}^{L}\sum_{j=1}^{M} sc_{ij}},\ j = 1, \ldots, M. \tag{2}$$

Table 4. The competences and the normalized average competence values

Competences	c_1	…	c_M
Normalized average competence values	c_1^{aver}	…	c_M^{aver}

Let us define functions $f_j : T_C \to \{c_j^{aver}, j = 1, ..., M\}$ that map the time tc_j devoted to teaching the competence c_j into the average level c^{aver} of acquirement of this competence. Thus

$$f_j(tc_j) = c_j^{aver} \quad \text{for each j=1,…, M.} \tag{3}$$

Let us assume that before studying the competences, the time spent on teaching is equal to 0, the level of the acquirement of this competence is also equal to 0 and after studying – the time spent on teaching the subject or subjects is equal to 1, the level of the acquirement of the competences is equal to 1, so

$$f_j(0) = 0 \ \text{ and } f_j(1) = 1 \text{ for each } j = 1,\ldots,M. \tag{4}$$

The family of functions which fulfill conditions (3)–(4) consists of many different kinds of functions. Let us choose quadratic polynomials:

$$f(t) = pt^2 + qt + r,$$
$$r = 0$$

because we have 3 points for approximation for each function. Namely in 0,1 and at the point tc_j (the example in Fig. 2).

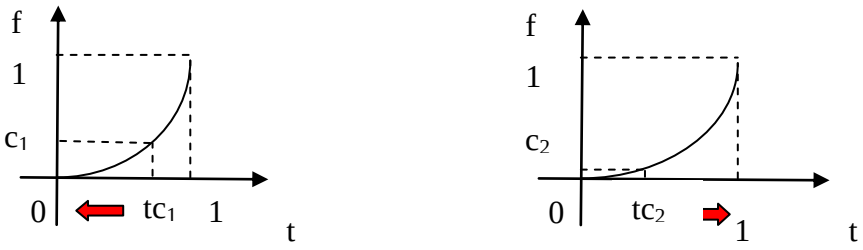


Fig. 2. Calculation of competence level and possible modifications

Next step is to change the distribution of time $tc_1,\ldots,tc_M$ according to the desired average competences c_j^*, namely to find the set $\{tc_1, \ldots, tc_M\}$ such that

$$\arg \min_{\{tc_1,\ldots tc_M\}} \sum_{j=1}^{M} \left(f_j(tc_j) - c_j^* \right)^2 \tag{6}$$

Let us consider the situation when all of the desired competences have the same weight and signification, so

$$c_j^* = \frac{1}{M} \text{ for all j = 1, …, M.} \tag{7}$$

After solving the task (6) with the desired competences written in (7), the next step is to change the times from (1) using the inverse matrix, namely

$$T = R^+ \cdot T_C \tag{8}$$

where R^+ is the pseudoinverse matrix (comp. Ref. 2), where $R^+ = R'(RR')^{-1}$.

3. The example of the system

Let the matrix R_{pc} and R_{sp} be defined as follows:

Table 5. Values of the matrix R_{pc}

Competence / Subjects	c_1	c_2	c_3	Times for the subjects	
p_1	0.7	0.3	0	t_1	0.3
p_2	0.1	0.8	0.1	t_2	0.2
p_3	0.1	0	0.9	t_3	0.2
p_4	0.2	0.2	0.6	t_4	0.3

Table 6. Values of the matrix R_{sp}

Subjects / Students	p_1	p_2	p_3	p_4
s_1	1	0.9	0.9	0.8
s_2	0.8	0.9	0.6	0.7
s_3	0.5	0.6	0.5	0.7
s_4	0.4	0.8	0.4	0.6
s_5	0.7	0.6	0.5	0.8

Using the max-min composition we get the matrix R_{sp} which values are put in Table 7:

Table 7. Values of the matrix R_{sp}

Competences / Students	c_1	c_2	c_3
s_1	0.7	0.8	0.9
s_2	0.7	0.8	0.6
s_3	0.5	0.6	0.6
s_4	0.4	0.8	0.6
s_5	0.7	0.6	0.6

Task 1

Let the vector of desired levels of competences be equal C*=[0.7 0.7 0.6]. Then we have to solve the equation R_{sc} o S=C*, where S is not given. After using the Sanchez algorithm (comp. Ref. 3) we get

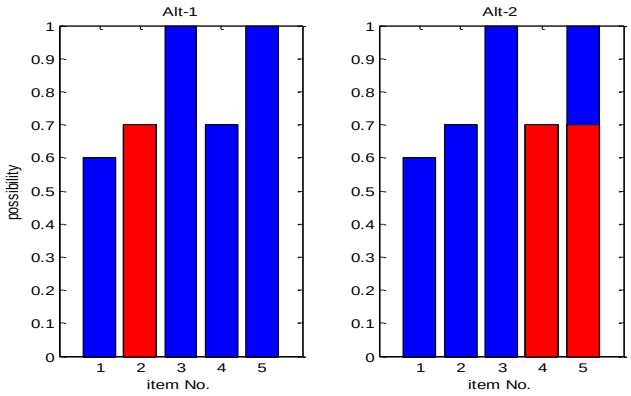


Fig. 3. The solution of the task 1

Variant 1 (Fig. 3, on the left) The possibility that students s_1, s_3, s_4 and s_5 fulfill the requirements C* belong to the intervals [0, 0.6], [0,1], [0,0.7], [0,1] respectively and in the case of the student s_2 it is equal to 0.7.

Variant 2 (Fig. 3, on the right) The possibility that students s_1, s_2, s_3 and s_5 belong to the interval [0,0.6], [0, 0.7], [0,1] and [0.7, 1] respectively and in the case of the student s_4 it is equal to 0.7.

The interpretation of the result can be as follows.

If we use variant 1, it is enough that one person fulfills the demands of business C*=[0.7 0.7 0.6]. It can be acceptable if we trust that the student 2 can answer the requirements with level 0.7 and the student 5 answers the demands at the level not less than 0.7. Moreover other students have to fulfill these demands with values according to variant 2. Really, student 4 has a level of competence 1 with 0.4 but we need 0.7 from C*, and student 5 has 0.7 for the same competence. Student 5 has a level of competence 2 with 0.6 but we need 0.7 from C*, and student 4 has 0.8 for the same competence. If we use variant 2 it is necessary to engage two people for fulfilling the demands of business C*=[0.7 0.7 0.6]. It can be acceptable if we trust that the student 3 can answers

the requirements with level 0.7. Moreover other students have to fulfill these demands with values according to variant 1.

Student 1 is good also for the demands of business C*=[0.7 0.7 0.6]. But we do not choose them in our solutions because they are too good (we ask more precise answer on the request!). For instance, if business would have to minimize of number of employees, they prefer variant 1.

Task 2

Using the data from matrix R_{pc} (table 5) and the formula (1) we can calculate the vector TC = $\{tc_1, \dots, tc_M\}$, which in the considered case is equal to TC = $\{0.31 \;\; tc_2=0.31 \;\; tc_3=0.38\}$.

Using the data from the matrix R_{sc} (table 7) and the formula (2) we can calculate the average levels of acquirements of competences: $c^{aver}{}_1 = 0.3030$, $c^{aver}{}_2 = 0.3636$, $c^{aver}{}_3 = 0.3333$.

Now using the functions defined in (5) and conditions (3) and (4), namely $f(tc_1)= c^{aver}{}_1$, $f(tc_2)= c^{aver}{}_2$, $f(tc_3)= c^{aver}{}_1\, c_3$ we have the functions:

$$f_1(x) = 0.03327\ 33x^2+0.9673\ x;$$
$$f_2(x) = -0.2506\ x^2+1.251\ x; \qquad (8)$$
$$f_3(x) = 0.1982\ x^2+0.8018\ x.$$

We have got three competences in our example (M = 3). If we want all competences to be equally important, then for the functions (6) we have got

$$C^*=[1/3\ 1/3\ 1/3]. \qquad (9)$$

The solution of (6) with functions (8) and constants (9) is the vector

$$T_C=[0.34 \;\; 0.28 \;\; 0.38]. \qquad (10)$$

Now for finding T we use (7), (10) and (1). On the basis of data from the table 1 we can get the matrix R and then using the formula $R^+=R'(RR')^{-1}$ we can calculate the matrix R^+ and by the formula (8), we can calculate the new time distribution for subjects: $t_1= 0.3592$, $t_2 =0.1541$, $t_3 =0.2418$, $t_4 =0.2449$.

This distribution of time devoted to teaching subjects from the set S would be optimal for the desired level of the competences C*=[1/3 1/3 1/3] if we could have the same students next academic year. If we suppose that abilities of students cannot change drastically from year to year such proposition will improve the education process.

4. Conclusions

Preparing good HEI's curricula requires that there should be taken into account a lot of factors, for example: the mission, means and academic staff of the institution, the needs of the labour market, candidates' competences and so on. Thus even if the curriculum is good, it must be checked and improved the next academic year.

In real life, curricula have 50-80 competences, 30-50 subjects and sometimes hundreds of students. Because of that there is a lot of imprecise data to analyze so the expert system will be very useful, especially one based on artificial intelligence and fuzzy logic.

The expert system can help young people to find their career path, but from the HEI's point of view it is very important that the system can help to prepare better, more efficient curricula which better suits the job market and the means of the institution. How to distribute time for subjects and for teaching competences effectively is not easy and it depends on the students' knowledge and skills.

References

[1]. A. Sokolov and O. Molchanova, Fuzzy assessment of test results, (Proceedings of European Society for Fuzzy Logic and Technology EUSFLAT. – Aix-les-Bains, France 2011).
[2]. C. R. Rao and C. Radhakrishna Rao and Sujit Kumar Mitra, Generalized Inverse of Matrices and its Applications (New York: John Wiley & Sons, 1971).
[3]. E. Sanchez, Resolution of composite fuzzy relation equations (Information and Control 30, 1976).

For contacts:
Aleksandra Mreła
Faculty of Technics
Kujawy and Pomorze University in Bydgoszcz
M. Piotrowskiego St.12-14,
Bydgoszcz, 85-098
POLAND
E-mail: a.mrela@kpsw.edu.pl
www.kpsw.edu.pl

Oleksandr Sokołov
Department of Informatics
Nicolaus Copernicus University
ul. Grudziadzka 5
87-100 Torun, POLAND
E:mail: oleksandr_sokolov@yahoo.com
www.umk.pl
Faculty of Technics
Kujawy and Pomorze University in Bydgosz
M. Piotrowskiego St.12-14,
Bydgoszcz, 85-098
POLAND

УСКОРЕНО СОФТУЕРНО ИЗПЪЛНЕНИЕ НА КРИПТОГРАФСКИ АЛГОРИТЪМ AES

Пламен Й. Стоянов

Резюме: Блоков алгоритъм AES (Advanced Encryption Standard), е широко използван в криптографски приложения. Той притежава висока ефективност на различни платформи, от 8-битови до 64-битови микропроцесори. В статията се представя ускорено софтуерно изпълнение на 8-битов RISC микроконтролер, типично използван при Smart карти, безжични сензорни модули и др. Предложената програма извършва криптиране с AES на блок от данни за 0,3 ms и заема програмна памет по-малко от 1.2 kB.
Ключови думи: криптография, AES, микроконтролер, Smart карти.

Fast speed software implementation of AES cryptographic algorithm

Plamen I. Stoianov

Abstract: Advanced Encryption Standard (AES) block cipher system is widely used in cryptographic applications. It is extremely efficient on many diferent platforms, ranging from 8-bit microcontrollers to 64-bit processors. This paper presents fast software implementation to an 8-bit RISC microcontroller, typical for Smart Cards, sensor nodes etc. Performance of implemented AES algorithm is about 0,3 ms per key schedule and encryption with a code size of less than 1,2 kilobytes.
Keywords: cryptography, AES, microcontroller, Smart Card.

1. Увод

Задача на криптографията е преобразуване на открит текст (plaintext) в шифриран текст (ciphertext) чрез използване на криптографски алгоритъм и еднозначно възстановяване на оригиналния текст. Процесът на преобразуване в шифриран текст се нарича шифриране или криптиране (encryption), докато обратният процес на възстановяване е дешифриране или декриптиране (decryption). В двата процеса се използва ключ за криптиране и ключ за декриптиране. Ако тези два ключа съвпадат, алгоритъмът се нарича симетричен, или още алгоритъм със секретен ключ. Ако ключовете са различни, алгоритъмът е асиметричен, или още алгоритъм с публичен ключ [1].

Най-широко използваният симетричен алгоритъм е DES (Data Encryption Standart). Алгоритъмът преобразува 64-битов открит текст в 64-битов шифротекст. Ключът е 64-битов, от които 56 бита се използват директно в алгоритъма, а всеки 8-ми бит се препоръчва за проверка по четност при съхранение и обмен. Поради ниската размерност на ключа и проведените успешни атаки срещу DES, през 1997 г. NIST (National Institute of Standards and Technology) стартира процедура по замяна на този стандарт. През 2000 г. е избран алгоритъм Rijndael, разработен от белгийски криптографи [2]. През следващата година е стандартизиран като AES (Advanced Encryption Standart) чрез FIPS PUB 197 [3]. Този алгоритъм има ефективна структура на преобразуване, използваща основни аритметични операции. Досега няма данни за извършени успешни атаки срещу AES. Изключение е използването на съкратени варианти, т.е. по-малко на брой етапи от предвидените в стандарта [4]. Целта на всички софтуерни изпълнения е ускоряване на процеса на криптиране и декриптиране, при възможно по-малка по обем програма. За реализация на

AES, е избран микроконтролер PIC18 на фирмата MICROCHIP. Притежава матричен умножител и възможност за лесно организиране на таблици.

2. Изложение

Математическа основа на AES е теория на полето. Извършват се действия с полиноми в поле на Галоа GF (2^8). Една част от операциите се извършват с байтове, а останалата с 4-байтови думи. Данните се представят като полиноми със степен по-малка от 8 и коефиценти в GF(2). Междинните резултати при обработката се наричат състояние (State) и се представят като квадратна матрица от 4 реда и 4 колони. Всеки байт от състоянието се бележи със $s_{r,c}$, където *r* е реда и *c* е колоната от 0 до 3. Данните се обработват в 10, 12 или 14 еднотипни етапа - N_r, като броят им зависи от разрядността на ключа – 128, 192 или 256 бита. В зависимост от ключа стандартът определя три варианта: AES-128, AES-192 и AES-256. Всеки етап се състои от 4 различни стъпки - SubBytes, ShiftRows, MixColumns, AddRoundKey, и допълнителен за получаване на подключовете за всеки етап – ExpandKey. Алгоритъмът се изпълнява в следната последователност :

```
Input = State , Key
        AddRoundKey(State, Key)
                ExpandKey(Key)
For round = 1 to N_r - 1
        SubBytes(State)
                ShiftRows(State)
                MixColumns(State)
                AddRoundKey(State,Key)
                ExpandKey(Key)
End for
                SubBytes(State)
                ShiftRows(State)
                AddRoundKey(State,Key)
Output = State
```

Финалният етап е изнесен отделно, тъй като не се изпълнява смесване на колоните. В отделните стъпки и получаване на подключове се изпълняват следните действия:

- **SubBytes** – за получаване на изходната стойност, се извършват две отделни изчисления: мултипликативна инверсия и свързано преобразуване. Ако a(x) е входният полином, то мултипликативна инверсия $a(x)^{-1}$ се намира чрез $a(x).a(x)^{-1} \equiv 1 \bmod m(x)$. Използва се неприводим полином $m(x) = x^8 + x^4 + x^3 + x + 1$. При размерност на полето от (2^8) елемента е удачно предварително изчисление на стойностите и запис в паметта като таблица с размерност 256 байта. Това води до по-висока скорост на изпълнение и по-малък обем на използваната памет. При изпълнение на свързано преобразуване всеки бит на изходната стойност се определя от сума по mod2 между определени битове на входната стойност и зададена константа. При модулна инверсия се използва таблица, следователно нейните стойности може да се преизчислят според условията на свързаното преобразуване. По този начин двете операции се изпълняват само с едно индексиране на таблица. За процеса на декриптиране е необходима допълнителна таблица. За извличане от таблица е необходим предварителен запис в регистър TBLPTR, поради което за замяна на един байт са нужни 6 цикъла на микроконтролера. За един ред или стълб са необходими 24 цикъла, а за цялата матрица - 96 цикъла.

- **ShiftRows** – извършва се ротация на думи в отделните редове на матрицата. Нулевият ред за всеки етап остава без промяна, докато 1-ви, 2-ри и 3-ти ред от матрицата, се ротират

наляво, съответно на 1, 2 и 3 байта. Стъпките SubBytes и ShiftRows може да се обединят, при което се съкращават инструкциите за повторно четене и запис. В резултат общото време за изпълнение изисква 96 цикъла, колкото е при SubBytes.

- **MixColumns** – всеки стълб от матрицата се представя като 4 членен полином *s(x)* с коефициенти, които са елементи на полето GF (2^8). Новата стойност $s'(x)$ се получава след умножаване на *s(x)* със зададен в стандарта полином:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (1)$$

Произведението се редуцира по модул (x^4+1) т.е:

$$s'(x) = a(x).s(x) \bmod (x^4+1)\ \} \quad (2)$$

В процеса на декриптиране (InvMixColumns) се използва инверсният полином $a^{-1}(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$, получен чрез $a^{-1}(x).\ a(x) \equiv 1 \bmod (x^4+1)$.
Действията в (2), при известните стойности на *a(x)* (1) може да се изразят матрично :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} . \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$ където c = 0 ÷ 3 са колоните (3)

След умножение, новите стойности на колоната са:

$$\begin{aligned} s'_{0,c} &= 2.s_{0,c} \oplus 3.s_{1,c} \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &= s_{0,c} \oplus 2.s_{1,c} \oplus 3.s_{2,c} \oplus s_{3,c} \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus 2.s_{2,c} \oplus 3.s_{3,c} \\ s'_{3,c} &= 3.s_{0,c} \oplus s_{1,c} \oplus s_{2,c} \oplus 2.s_{3,c} \end{aligned} \quad (4)$$

Коефицентите на използвания полином се различават само с 1, поради което може да се използва зависимостта: $c.s_i \oplus (c+1).s_j = c.(s_i \oplus s_j) \oplus s_j$. Ако предварително се изчисли сумата $Temp = s_{0,c} \oplus s_{1,c} \oplus s_{2,c} \oplus s_{3,c}$, то стойността на първия байт от колоната може да се представи чрез $s'_{0,c} = (s_{0,c} \oplus s_{1,c}).2 \oplus s_{0,c} \oplus Temp$. Операциите по умножение се намаляват, при запазване броя на сумиранията по mod2. За намиране на четирите байта от колоната са необходими 32 цикъла, следователно за всички 16 байта на матрицата са нужни 128 цикъла.

- **AddRoundKey** – всеки байт от матрицата се сумира по mod2 със съответния байт от текущия подключ, за получаване на новото състояние. Изисква се прочитане и сумиране по mod2, което отнема 32 цикъла за всичките 16 байта. В тялото на цикъла изпълнението на AddRoundKey е непосредствено след MixColumns и следователно двете стъпки може да се обединят. След получаване на новата стойност от смесване на колоните, тя се сумира със съответния байт от ключа и се записва в матрицата като ново състояние. В резултат на обединението на двете стъпки се съкращават инструкциите за четене, при което общото време за изпълнение от 128 + 32 цикъла се намалява на 128 + 16 = 144 цикъла.

- **ExpandKey** – за всеки отделен етап се използва различен ключ, получен от основния чрез преобразувания. Както и при данните, областта на ключа се представя като матрица от 4 реда и 4 стълба. След всяко изпълнение на AddRoundKey (с изключение на последния етап), четвъртият ред се ротира наляво на един байт, и се извършва SubBytes по същата таблица, както и за данните. Първият байт се сумира с константа, която е различна за отделните етапи. Следва сумиране по mod2 на четвърти към първи, първи към втори, втори към трети, и трети

към четвърти ред. По този начин се получават подключовете за всеки следващ етап. Необходимото време за изпълнение на всяка стъпка, е 53 цикъла. Те могат да се намалят ако замяната на байтове се извършва от таблица в данновата памет, вместо от програмната памет. Друг начин за ускоряване е чрез изнасяне на ExpandKey извън обработката на данни. При всяка смяна на потребителския ключ се извършва предварително изчисление на всички подключове. Стойностите се записват в оперативната памет, и в процеса на криптиране само се прочитат, без да се изчисляват. Много блокове се криптират с един и същ ключ, при което общото време за изпълнение се намалява. Недостатък е заемане на допълнителни 160 байта от оперативната памет за всички подключове.

Декриптирането е в обратен порядък. Първо се извършва ротация на редове надясно (при криптиране е наляво), след което се заменят байтовете според инверсна таблица. Възможно е да се използва една и съща таблица, но това би довело до увеличаване на времето за декриптиране. Търсенето на съвпадение е по-бавно и зависимо по време от входните данни. По-високо бързодействие се постига чрез добавяне на нова таблица и увеличаване на програмната памет допълнително с 256 байта. Друга разлика е използването на инверсен полином с коефициенти различни от коефицентите на полинома при криптиране.

3. Резултати

Таблица 1 Сравнителни резултати за изпълнение AES-128

	необходим брой цикли			обем програма (bytes)
	предварителни изчисления	криптиране	общо	
(1) Daemen Rijmen [5]	-	3744	3744	826
	-	3168	3168	1016
(2) Rinne [6]	-	3766	3766	3410
(3) Poettering [7]	756	2474	3230	1818
(4) Permadi [8]	-	3137	3137	892
	933	2611	3544	1130
(5) получени резултати	-	2873	2873	1185
	782	2341	3123	1159

Получените резултати (5) при софтуерно изпълнение на криптиране с алгоритъм AES-128, са приведени в таблица 1. Извършено е сравнение на необходимите брой цикли за изпълнение с известни решения. В третата колона са показани необходимият брой цикли за изпълнение процеса на криптиране. Това е основен показател, тъй като определя скоростa на обработка на данните. В четвърта колона е общият брой цикли. Има варианти с предварително изчисление на подключовете, след което криптирането се извършва с по-висока скорост. Това предварително изчисление се извършва само при смяна на потребителския ключ, след което множество блокове се криптират с изчислените подключове. Недостатък е заемането на допълнителни 160 B от данновата памет. В пета колона е необходимият размер на паметта за разполагане на програмата. Предложение (1) е

на създателите на алгоритъм AES, разработено на основата на микроконтролер на INTEL. Вторият вариант е оптимизиран за по-бързо изпълнение на криптирането. Предложения (2) и (3) са на основата на микроконтролери AVR на фирмата ATMEL. В [7] са разработени три варианта при различен брой цикли и размерност на програмата. В таблицата е представен вариантът с най-млък брой цикли. Предложения (4) и (5) са на основата на микроконтролери PIC18 на фирмата MICROCHIP. Разработената програма (5) извършва криптиране на 16 байтов блок за по-малко цикли в сравнение с (4), основно поради по-бързото изчисление на полиноми в стъпка MixColumns. Друга причина за по-високо бързодействие е неизползването на подпрограми. Те намаляват обема на програмата, но изискват допълнително 4 цикъла при изпълнение. Възможност за намаляване на необходимия брой цикли е чрез пренасяне на таблицата за замяна в данновата памет. Особеност на използвания микроконтролер е, че извличане на байт от програмната памет изисква 5 цикъла, докато индексно четене от данновата памет се изпълнява за 3 цикъла. Таблицата за замяна се използва 16 пъти в SubByte, което предполага 320 цикъла по-малко при варианта с предварителни изчисления. Постигнатото съкращаване на цикли при криптиране е за сметка на увеличаване на програмната памет със 768 B и предварителните изчисления с 512 цикъла.

4. Заключение

В статията се разглежда широко използваният симетричен алгоритъм AES. Анализирани са отделните стъпки при криптиране, с цел намиране на възможности за по-бързо изпълнение. Предложеното софтуерно решение е сравнено с други известни разработки. Чрез изпълнение на симулатор е доказана работоспособността на програмата, както и необходимия брой цикли на микроконтролера.

Литература

[1]. Антонов, П., С. Малчев, Криптография в компютърните комуникации, Варна, 2000.
[2]. Daemen, J., V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission, 1999, http://csrs.nist.gov
[3]. FIPS PUB 197, Advanced Encryption Standart (AES), Federal Information Processing Standards (FIPS), NIST, US Departmaent of Commerce, 2001.
[4]. Biryukov, A., O.Dunkelman, N.Keller, D.Khovratovich, A.Shamir, Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds, EUROCRYPT 2010, LNCS 6110, p.p.299-319.
[5]. Daemen, J., V. Rijmen, Efficient Block Ciphers for Smartcards, .1 st USENIX Workshop on Smartcard Technology, 1999.
[6]. Rinne, S., T. Eisenbarth, C. Paar, Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers, Software Performance Enhancement for Encryption and Decryption (SPEED 2007) 2007.
[7]. Poettering, B., AVRAES : The AES block cipher on AVR controller, 2007
[8]. http://point-at-infinity.org/avraes/
[9]. Permadi, E., Fast AES Implementation on PIC18F4550 ,2008.
[10]. http://edipermadi.wordpress.com/2008/06/17/fast-aes-implementation-on-pic18f4550/

За контакти:
гл. ас. инж. Пламен Й. Стоянов
катедра „Комуникационна техника и технологии”
Технически Университет-Варна
E-mail: pl_stoianov@tu-varna.bg

ПРОГРАМНА БИБЛИОТЕКА ЗА АВТОМАТИЧНО КОМБИНИРАНЕ НА ИЗОБРАЖЕНИЯ

Юлка П. Петкова

Резюме: Автоматичното комбиниране на изображения с припокриващи се области, така че да се формира „безшевно” панорамно изображение е проблем, който е особено актуален, с голяма сложност и голяма необходимост. Липсата на отворени кодове на популярните програми за такова комбиниране прави невъзможно тяхното използване за изследователски цели. В статията се описва създадена програмна библиотека за нуждите на подобни обработки. Алгоритмите, включени в библиотеката, са с голяма надеждност и в същото време достатъчно бързи. Компонентите в библиотеката са с модулна организация. Библиотеката е крос-платформена, преносима и високо ефективна, което я прави удобна за решаване на широк кръг задачи от областта на обработката на изображения. Получените резултати потвърждават точността, бързодействието и надеждността на алгоритмите, включени в библиотеката.
Ключови думи: обработка на изображения, панорамни изображения, динамична библиотека, крос-платформеност

Dinamic Link Library for Automatic Image Stitching

Yulka P. Petkova

Abstract: Automatic combining of images with regions of overlap in order to form the "seamless" panoramic image is a very important problem, with a greater complexity. The lack of popular open source programs for such combination makes it impossible to be used for research purposes. The article describes a dynamic link library (DLL) created for the needs of such treatments. The included in the library algorithms have a greater reliability and at the same time they are fast enough. The components included in the library have a modular organization. The library is cross-platform, portable and highly efficient, which makes it suitable for a wide range of tasks in the field of image processing. The obtained results confirm the accuracy and reliability of algorithms included in the library.
Keywords: Image processing, Automatic Image stitching, Panoramic Images, Dinamic Link Library, Cross-platform

1. Въведение

Съвременната компютърна визия е необходима и бързо развиваща се област поради все по-многобройните и по-разнообразни области на приложение. Една от задачите, които се налага да бъдат решавани за удовлетворяване на предизвикателствата на различните приложения на компютърната визия, е задачата за автоматично комбиниране на изображения с припокриващи се области, така че комбинацията им да формира панорамно изображение с минимални следи от обединяването (т. нар. „безшевно” обединяване). Сложността на проблема нараства, когато изображенията са от различни типове сензори, с различна степен на осветеност и шум, с триизмерни геометрични трансформации като транслация, ротация, мащабиране и перспективни изкривявания. Най-често изображенията, които трябва да се обединят в панорамно, са получени от модерни цифрови фотоапарати с високи разделителни способности, поради което те съдържат и голям обем от данни. Обработката на тези данни трябва да бъде точна и максимално бърза.

Известни са програмни приложения, които създават панорами, използвайки серия от входни изображения, но най-често тези приложения изискват потребителска намеса – потребителят на приложението маркира общите точки, области, обекти за различните

изображения. Такова поведение на програмното приложение не може да се определи като автоматично. Съществуват и малко на брой разработки, които успяват да комбинират входните изображения с общи области в панорамно изображение, без да изискват човешка намеса.

Всички разработки от последния вид са със затворен код. Голяма част от алгоритмите, които се използват в тези приложения, или са патентовани, или въобще не са публикувани. Използвайки тези алгоритми, са разработени комерсиални приложения, които са широко известни, особено сред фотографите.

Най-известните и утвърдени приложения са следните:

ArcSoft Panorama Maker Pro 5 [1] – едно от най-разпространените приложения от този тип. Развива се активно от 2005 година. Съвместимо е с различните версии на операционните системи Windows и Mac OS. Поддържа хоризонтални, вертикални комбинирани и 360 градусови панорамни изображения. Тази програма е основен продукт на софтуерната компания, която я разработва.

PTgui Pro 9.0.4 [2] – тази програма представлява графичен интерфейс към библиотеката *Panorama Tools* [7] на професор Helmut Dersch. Самата библиотека *Panorama Tools* е със затворен код (има части, за които кодът е отворен, но те не са от съществено значение за комбинирането на изображения). Приложението има каества, подобни на *ArcSoft Panorama Maker Pro 5* и отново е съвместимо с версиите на операционните системи Windows и Mac OS.

Kolor auto pano pro 2.5 [3] – софтуерна система за създаване на панорамни изображения, създадена през 2006 година. Тази програма има версии за Linux и iOS. Основен продукт е на компанията, която я разработва.

AcroPano Photo Stitcher 2.1 [4] – единствен продукт на компанията разработчик и се развива от 2005 година. Удобен за ползване, с изчистен графичен интерфейс, но отстъпва по качество на по-горните приложения. Има версия само за Windows.

Panorama Builder 7.0 [5] – тази система предлага изцяло Web базиран интерфейс за работа. Самото приложение, което извършва комбинирането, е Windows базирано. Качеството на панорамите е сравнително добро, но често се наблюдават грешни слепвания, водещи до появата на дефекти в изходното панорамно изображение.

The Panorama Factory 5.3 [6] – приложение за Windows и Mac OS платформи. Отстъпва и по качество, и по бързодействие на горните продукти. Потребителският интерфейс не е много усобен за използване от начинаещи потребители. Липсва ръчен режим за настройване.

Освен програмите, описани горе, съществуват и други, но техните ограничения са значително по-големи. Има такива програми, които изискват от потребителя задължително да селектира минимален брой от съответстващи точки от входните изображения. Затова тези програми не могат да се класифицират като успешно решаващи задачата за автоматичното комбиниране на изображения в панорами. При автоматичното комбиниране входните изображения трябва да могат да се „зашиват” без нуждата от потребителска намеса.

Фактът, че има софтуерни компании, чиято основна дейност е свързана с автоматично комбиниране на панорамни изображения и основните им продукти (програмни приложения) са насочени в това направление, показва сериозната значимост и голямата мащабност на тази задача.

На този етап няма безплатно приложение, което ефективно да решава проблема за автоматино комбиниране на изображения за получаване на панорамно изображение. Всички съществуващи комерсиални приложения се развиват непрекъснато с цел повишаване на скоростта, подобряване на точността, усъвършенстване на сливането (създаване на незабележим „шев”), за да се избегнат дефекти в крайното панорамно изображение.

Скоростта на работа на алгоритмите е критична, защото се обработва голям обем от данни (снимките, генерирани от съвременните цифрови фотоапарати, се характеризират с

висока разделителна способност) и затова се търсят начини и стратегии за ускоряване на процеса на комбиниране на изображения. За всеки отделен етап от тази задача могат да се използват различни стратегии и подходи за ускоряване [9], [10]. За да могат да се прилагат тези стратегии обаче, трябва подробно да се анализират особеностите на всеки основен етап от комбинирането на изображения.

2. Архитектурни особености на програмната библиотека за комбиниране на изображения

Програмната библиотека включва алгоритми и структури от данни и е проектирана така, че всеки модул, включен в нея, да е преносим и преизползваем. Това позволява използването ѝ с различни платформи и за най-различни задачи от областта на компютърната визия. Една от задачите, които могат да бъдат успешно решени чрез използването на библиотеката, е задачата за комбиниране на изображения.

Основните усилия са насочени към йерархията и софтуерния дизайн на библиотеката, така че нейните структури от данни, алгоритми и стратегии да бъдат подходящи за използване в различни задачи от областта на компютърното зрение и в частност цифрова обработка и анализ на изображения. Библиотеката не е предназначена само за автоматично обединяване на изображения. Тя може да бъде използвана в решаването на широк спектър от задачи, свързани с цифровата обработка и анализ на изображения. Като допълнителни свойства библиотеката поддържа системно независими, преносими модули за многонипкови паралелни изчисления, зареждане и представяне на различни формати изображения, алгоритми за всеки етап от процеса на автоматично обединяване на изображения, използващи оптимално апаратните ресурси на всеки компютър.

Отговорността при концептуалното проектиране на програмни библиотеки е по-голяма в сравнение с проектирането на приложения, защото за разлика от дадено програмно приложение, целта на библиотеката е да бъде използвана в различни приложения. Това означава, че ако в дадена нова версия на една програмна библиотека настъпят промени в интерфейсите и начина на връзка с външните приложения, то приложенията, които работят с нея, стават несъвместими с тази нова версия. При планирането и проектирането на програмната библиотека тези факти са отчетени.

2.1 Основни изисквания към програмната библиотека

Преди етапа на проектиране са формулирани следните основни изисквания към библиотеката:

- ✓ Крос-платформеност – библиотеката трябва да може да се използва под различни платформи и широк спектър от операционни системи. Задължителните платформи, които трябва да се поддържат, са Microsoft Windows, Unix/Linux и Mac OS.
- ✓ Преносимост – кодът на библиотеката да включва само стандартни и преносими компоненти, така че да може да се компилира и за алтернативни платформи за мобилни устройства като Android, Symbian, Windows CE, Windows Phone и др. Това изискване категорично забранява функционалността на библиотеката да е зависима от каквато и да е допълнителна помощна библиотека, специфична за дадена платформа или платформи.
- ✓ Възможност за използване под различни форми – това изискване има за цел да гарантира, че библиотеката може да се компилира като статична, динамична или като изпълним файл, така че нейните потребители да имат възможност да изберат най-удобния за тях вариант.

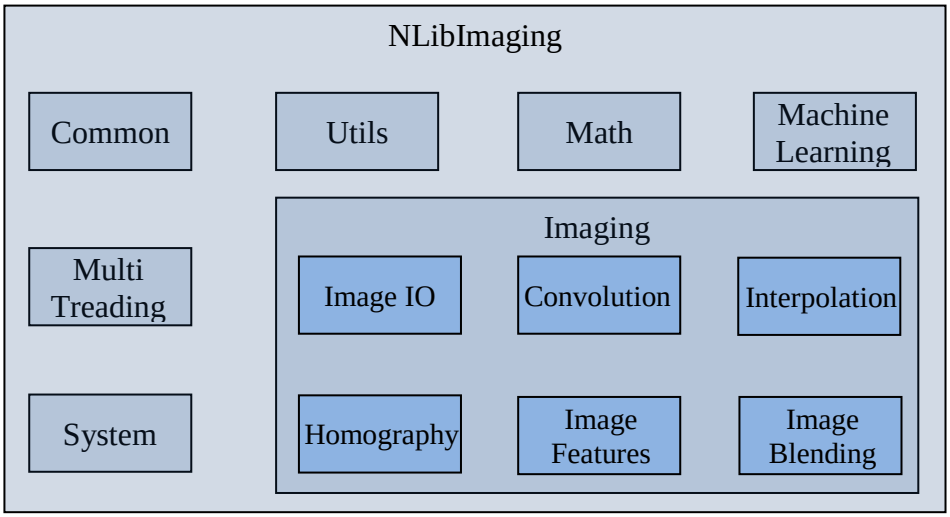
- ✓ Предоставяне на богат набор от средства за успешно решаване на всеки един от етапите на задачата за автоматично комбиниране на изображения в панорами. Тези средства трябва да бъдат максимално преизползваеми и по възможност да могат да се използват и в други приложения, решаващи различни задачи, в които биха били полезни.
- ✓ Предоставяне на допълнителни средства с общо предназначение за използване в широк набор от задачи за цифрова обработка и анализ на изображения.
- ✓ Разпределение и организиране на алгоритмите и структурите от данни в подходящи модули, съобразени с техните общи характеристики, с цел по-лесна поддръжка и разширяване на библиотеката.
- ✓ Възможност за работа със системно-зависими компоненти през системно-независими интерфейси. Тези компоненети често са свързани с характерните особености на конкретната платформа, за която е компилирана библиотеката. Целта е да може да се достъпи до тези компоненти през системно-независими интерфейси.
- ✓ Предоставяне на примитивни структури от данни за описание на геометрични компоненти, математически функции, изображения и др.
- ✓ Преносим модул за многонишкова паралелна обработка, който може да бъде използван както от вътрешните алгоритми на библиотеката, така и от външни потребителски алгоритми.
- ✓ Реализация на различни версии на алгоритмите, оптимизирани за бързодействие и за намаляване на изискванията към необходимата оперативна памет.
- ✓ Консистентност в софтуерния дизайн, който да минимизира зависимостите между отделните модули в библиотеката и да осигури подходящи интерфейси за лесно ползване и разширяване на всеки един от модулите.
- ✓ Предоставяне на алтернативни алгоритми за всеки един от етапите на автоматично обединяване на изображения.
- ✓ Осигуряване на минималистични интерфейси за максимално лесно използване на библиотеката.
- ✓ Осигуряване на опционални допълнителни интерфейси за фина настройка на всеки алгоритъм.

2.2. Основни модули на програмната библиотека

Едно от изискванията, представени в предходната точка, е свързано с модулната организация на библиотеката. Тази организация позволява гъвкавост при организиране на зависимостите и разширяване на библиотеката. Всеки модул се формира от съвкупност от класове, които са сходни или допълващи се по функционалност. Например, всички класове за избор на признаци от входни изображения (първи етап), заедно със структурите от данни, описващи признаците, формират даден модул, а класовете „зашиване” (четвърти етап) формират друг модул. Следващата фигура илюстрира основните модули на библиотеката.

С *Common* са обозначени всички примитивни типове и структури от данни. Те са общи за всички модули и могат да бъдат използвани от всички компоненти и класове. Решението за използване на такива типове и структури е мотивирано от изискванията за кросплатформеност, преносимост, единност на интерфейсите и пр. Например структурите за четириъгълник, точка, линия, вектор, елипса и др. могат да бъдат използвани от модула за математически изчисления, модула за обработка на изображения и др.

Модулът за математически изчисления *Math* е предназначен да съдържа контейнери и алгоритми за работа с матрици, в частност матрици за работа с трансформации.



Фиг. 1. Основни модули на библиотеката NLibImaging

Модулът с алгоритми и помощни класове от общ характер *Utils* съдържа помощни класове и алгоритми с общо предназначение като генератори на случайни числа, таймери и други.

Модулът *Machine Learning* е предназначен да съдържа алгоритми за машинно обучение. Такъв е алгоритъмът RANSAC.

Multi Threading е модул за единен интерфейс за работа с нишки под различни платформи. Модулът съдържа и средства за разпаралелване на съществуващи алгоритми и осигурява базови класове за еднонишкови задачи.

System е модул, осигуряващ общ интерфейс за достъп до специфични системни характеристики и ресурси за различни платформи. Класовете в този модул са проектирани с цел получаване на полезна системана информация като броя на процесорните ядра, общото количество оперативна памет, свободната памет и др. Информацията, свързана с паметта, може да бъде изпозлвана за реализация на механизъм за кеширане при недостиг на памет. Този механизъм е много полезен, когато входните изображения са много на брой и големи по размер. Липсата на такъв механизъм може да доведе до изчерпване на свободната оперативна памет, а оттам и до проблеми с производителността и дори до неуспех на опитите за алокация на памет за резултатното изображение.

Модулът за цифрова обработка и анализ на изображения *Imaging* е най-голям и включва структури за представяне на изображения от различен тип, алгоритми за обработка и анализ на тези изображения и допълнителни вътрешни модули за конкретни задачи. Част от тези модули съответстват на основните етапи от задачата за автоматично комбиниране на изображения. Модулът *Imaging* има достъп до функционалността на всички други модули, представени дотук. Типовете и структурите от данни са унифицирани и това прави комуникацията между модулите гладка и надеждна. Този модул включва в себе си сопълнителни модули за решаване на конкретни проблеми и задачи, като: модул за входно-изходни операции над изображенията (*ImageIO*), модул за интерполационни операции над изображения (*Interpolation*), за конволюционни операции над изображения (*Convolution*), за избор, описание, оценка и съпоставяне на прознаци (*Image Features*), за изчисляване на хомография (*Homography*) и за „зашиване” на изображенията в изходна панорама (*Image Blending*).

За реализация на библиотеката е използван програмният език C++.

За да се тестват и демонстрират възможностите на библиотеката и нейната способност успешно да решава задачата за автоматично комбиниране на панорамни приложения, са създадени две приложения – едното е конзолно, а другото е с графичен интерфейс. Не са

забелязани проблеми с преносимостта на библиотеката. Получените резултати са много добри както по отношение на бързодействието, така и по отношение на качеството на получените комбинирани панорамни изображения.

3. Заключение

С цел разработване и изследване на алгоритми за автоматично комбиниране на изображения с общи части в панорамни е проектирана и разработена програмна библиотека, която също може да се използва и при решаването на широк набор от задачи от областта на цифровата обработка и анализа на изображения. Към библиотеката са определени и допълнителни изисквания като крос-платформеност, преносимост, висока ефективност, надеждност, преизползваемост и други. Компонентите в библиотеката са с модулна организация, като отделните модули са организирани в собствени именувани пространства. Реализирани са модули за матрични математически изчисления и машинно обучение, които могат да се използват и от външни програми и библиотеки при решаването на широк обхват от задачи. Разработен е многофункционален модул за цифрова обработка и анализ на изображения. Реализирани са различни алгоритми за всеки основен етап от задачата за автоматично комбиниране на изображения. Всички тези алгоритми са представени чрез йерархия от класове, което позволява те да имат общи интерфейси и да могат да се преизползват. Алгоритмите са оптимизирани по отношение на бързодействие и надеждност с използването на различни стратегии и на алтернативни идеи [8].

Литература

[1]. http://www.arcsoft.com/estore/software_title.asp?ProductCode=PMK5PRO
[2]. http://www.ptgui.com
[3]. http://www.kolor.com
[4]. http://www.acropano.com
[5]. http://www.panobuilder.com/index.asp
[6]. http://www.panoramafactory.com
[7]. http://panotools.sourceforge.net
[8]. Petkova Yulka P., Nuri Nuri, An algorithm for fast image registration and feature matching for the purposes of image stitching, CompSysTech '11 Proceedings of the 12th International Conference on Computer Systems and Technologies, ACM New York, NY, USA ©2011, ISBN: 978-1-4503-0917-2 doi>10.1145/2023607.2023647, Pages 228-233
[9]. Petkova Yulka P., Todorka Georgieva, "Image Stitching – Basic Problems and Approaches for their Solution", Nis, Serbia XLVI International Scientific Conference on Information, Communication and Energy Systems and Technologies – ICEST2011, June 29th- July 1st 2011, Nis, Serbia, Proceedings of papers Vol. 3, pp. 709 – 712
[10]. Петкова Ю., „Методи за създаване на панорамни изображения", Дни на науката'2011, Велико Търново, 20-21 май 2011, ISSN 1314-2283, Том 2, стр. 350 – 359

Благодарности

Авторът изказва своята искрена благодарност към инж. Нури Нури за неговото активно съдействие при проектирането и разработването на библиотеката.

За контакти:
гл. ас. д-р Юлка П. Петкова,
катедра „Компютърни науки и технологии",
Технически университет – Варна,
e-mail: yulka.petkova@tu-varna.bg.

ГЕНЕТИЧЕН АЛГОРИТЪМ ЗА РАЗПОЗНАВАНЕ И ЛОКАЛИЗИРАНЕ НА МНОЖЕСТВО ЕТАЛОННИ ИЗОБРАЖЕНИЯ В ПО-ГОЛЯМО ИЗОБРАЖЕНИЕ

Юлка П. Петкова

Резюме: Статията разглежда проблема за търсене на множество еталонни изображения в по-голямо изображение. За целите на търсенето се прилага генетичен алгоритъм. Като мярка за подобие се използва нормализираната взаимна корелация, която се явява ценовата (фитнес) функция на генетичния алгоритъм. Предложени и тествани са два подхода – за последователно търсене на единично еталонно изображение и за едновременно търсене на всички еталонни изображения, чийто брой е предварително известен. Експериментите са проведени върху изображения на печатни платки с монтирани върху тях електронни елементи. Получените резултати показват достатъчно добра точност на локализацията, както и достатъчно добро бързодействие, за да може да се препоръча използването на тази техника в автоматизирането на качествената инспекция на електронни изделия, както и за други цели, изискващи откриването на множество еднотипни изображения в по-голямо изображение.
Ключови думи: Регистриране на изображение, Локализиране на еталон, Генетичен алгоритъм, Съответствие на точки, Инспекция на печатни платки

Genetic Algorithms for Detection and Localization of Multiple Reference Images in a Larger Image

Yulka P. Petkova

Abstract: The paper considers the problem of searching for multiple template images in a larger image. Genetic algorithm is used for the purpose of searching and localization. The normalized cross-correlation is used as a measure of similarity, i. e. it is the cost (fitness) function of the genetic algorithm. There were two types of searching - exhaustive and simultaneous (parallel) searching of the template. Experiments were carried out on images of printed circuit boards with mounted electronic components. The obtained results show sufficient precision of localization, and good enough run-time to be able to recommend the use of this technique in automated quality inspection of electronic products and for other purposes requiring registration of multiple identical template images in a greater image.
Keywords: Image registration, Template matching, Genetic algorithm, Point matching, PCB inspection

1. Въведение

Генетичните алгоритми (ГА) използват механизма на естествения подбор и биологичната генетика [10] и са изчислителни модели на естествената еволюция, в която по-силните физически индивиди са по-склонни да бъдат победители в конкурентна среда. ГА са вероятностни алгоритми за търсене на популационно ниво. Обикновено ГА се прилагат за решаване на оптимизационни проблеми. Предметната област на ГА включва задачи от комбинаториката, биоинформатиката, теорията на игрите и др., но ГА се прилагат също и за обработка на изображения и разпознаване на образи [1 - 9].

Същността на ГА се състои в това, че търсенето на решение се извършва в подмножество от точки от пространството за търсене. Това се постига чрез създаването на множество от потенциални решения, което формира популация. Кодирането на данните определя начина на представяне на потенциалното решение. Предполага се, че

потенциалното решение може да се представи във вид на параметри (гени), които могат да се съединят в прости структури от данни (хромозоми).

Популацията се усъвършенства с помощта на генетични оператори, отговарящи за изменчивостта, и ценова (фитнес) функция, която моделира естествения подбор. Ценовата функция служи за оценка на пригодността и приспособимостта на съответната хромозома. За конкретен тип задачи обикновено се прилагат установени ценови функции.

Наследствеността се обезпечава от това, че новите хромозоми се формират от хромозомите на предходното поколение и съответно имат общи гени с него. За правилната работа на ГА е много важно подходящо да бъдат кодирани данните и да бъде избрана ценовата функция. Ако ГА е правилно реализиран, то с всяко ново поколение средното значение на фитнес функцията на популацията, както и нейното най-добро значение се увеличават и се стремят към глобалния оптимум.

В областта на обработката на изображения ГА могат да се използват както при подготовката на изображенията за разпознаване (предварителна обработка), в частност при филтрация [1, 3], сегментация, анализ на сцени [1, 2], така и непосредствено за целите на разпознаването [5, 9] или регистрацията [4, 6, 7, 8]. Регистрацията е една от задачите в обработката на изображения, която се решава при съпоставяне на две изображения, направени по различно време, от различни датчици или от различни гледни точки. Регистрацията на изображения се определя и като съпоставяне на обектите в една и съща сцена. Тя се прилага в редица научни области, включително в анализ на медицински изображения, компютърно зрение и разпознаване на образи. Основното предимство на подхода с изпозлване на ГА за регистрация на изображения е, че не се налага предварително „изравняване" на изображението, за да се гарантира получаването на добри резултати [6].

2. Приложение на ГА за локализиране на множество еталонни изображения в по-голямо изображение

ГА стартира със създаването на случайна популация от решения и ценова или фитнес стойност, която се изчислява за всеки член на популацията. За целите на съпоставянето с еталон едни от най-подходящите мерки, които могат да се използват като ценови стойности, са тези на нормализираната взаимна корелация и на корелационния коефициент. Използвайки стойностите на фитнес функцията като критерий за устойчивостта и качествата на съответния индивид от поколението, се избират родители. Тези родители след това се комбинират, за да се генерира нов набор от решения, наречени потомство или поколение. Този нов комплект от решения, ако имат добра фитнес функция, се използва за замяна на некачествените членове на първоначалната популация. Процесът на създаването на поколение, изчисляване на ценовата стойност и изборът на устойчиви индивиди се повтаря, следвайки естественото еволюционно развитие, за да доведе до по-добри приближения, а оттам и до оптималното решение.

Операторът за мутация се използва за създаване на нови решения чрез произволна промяна в едно решение (хромозома).

Генетичният алгоритъм търси в ограниченото пространството на пикселите, които принадлежат на ръбовете в изображението. Той се осъществява с помощта на популация от индексите, определящи координатите на точките от ръбовете и отделна популация на индексите, определящи ъгъла на завъртане на еталона, както и неговия размер. Когато се прави инспекция на електронните елементи върху една печатна платка например, се има предвид, че обикновено резисторите, кондензаторите и други елементи се разполагат успоредно на едната или на другата страна на платката, т. е. те са завъртяни на ъгъл 0^o, 90^o, 180^o или 270^o. Недостатък на използването на само четири ъгъла е, че действителният ъгъл на въртене понякога не е определен. Електронните елементи от един и същ тип могат да

варират и по големина. За повишаване на ефективността на ГА се предполага, че елементите върху печатните платки могат да имат четири различни размера, определени от съответния мащабен коефициент, приложен върху еталонното изображение. Предимството на използването на четири ъгъла, както и на четири мащабни коефициента е, че изчислителната сложност се поддържа ниска. За всеки индивид (т. е. координати на пиксел от ръба, ъгъл и мащабен коефициент) се изчислява фитнес стойност чрез изчисляване на стойността на взаимната корелация (NCC) или на корелационния коефициент (NCCE). От текущата популация се избират двама родители, от които, чрез линейна рекомбинация (1), се генерират двама наследници.

$$\text{Наследник 1} = a*\text{родител\_1} + (1 - a)*\text{родител\_2}$$
$$\text{Наследник 2} = a*\text{родител\_2} + (1 - a)*\text{родител\_1} \quad (1)$$

където a е случайно число в интервала между 0 и 1.

Ново потомство може да се генерира и чрез кръстосване (едноточково или двуточково) по механизъм, показан на фигура 2 а и б.

а) Едноточково кръстосване

б) Двуточково кръстосване

Фиг. 1. Кръстосване – а) едноточково; б) двуточково

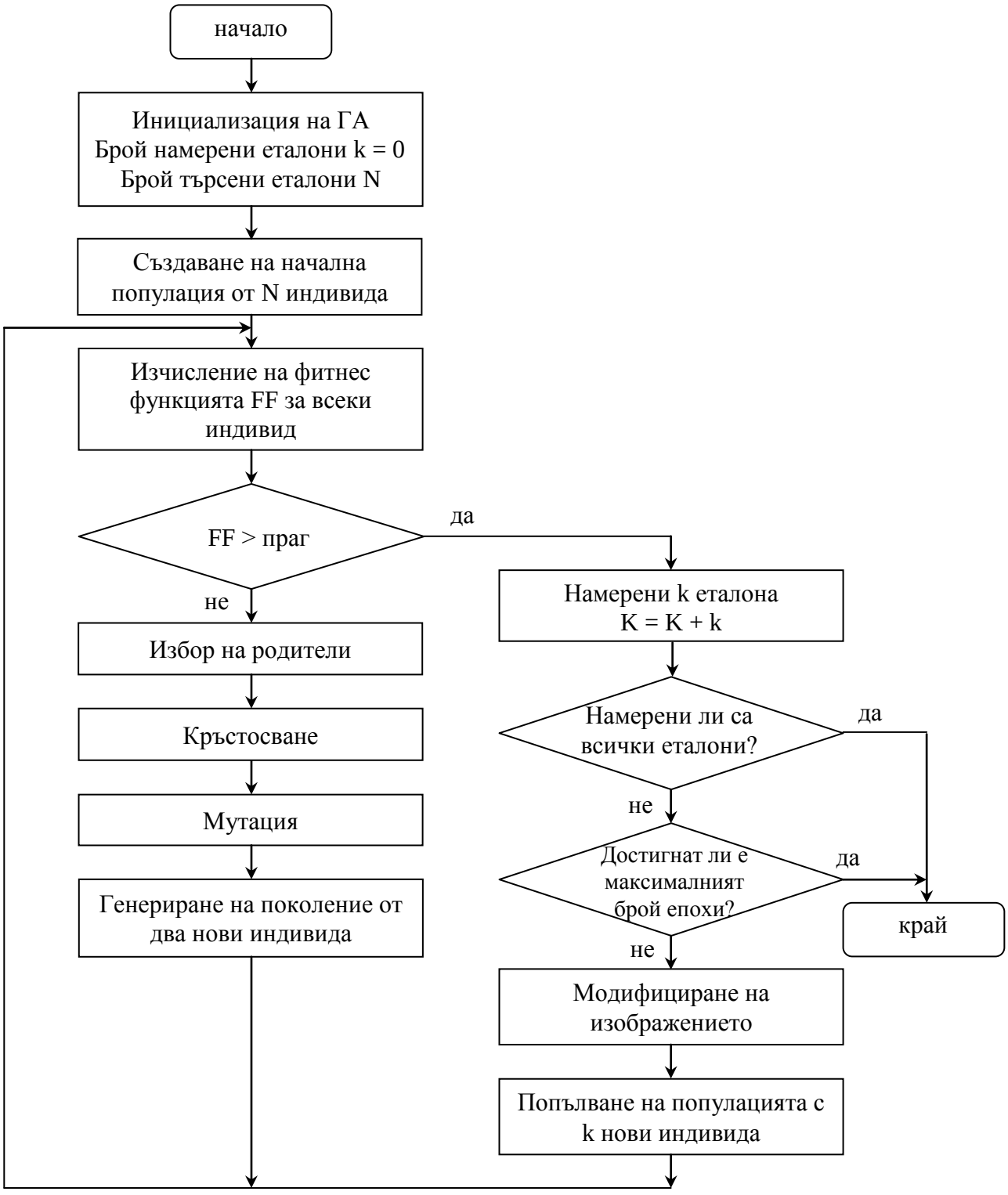
Мутацията в гените на индивидите се извършва само в последните 4 бита на съответния ген, в които с двоични комбинации са кодирани четирите възможни ъгъла на ротация и четирите възможни размера на еталонното изображение.

По принцип един генетичен алгоритъм е добър в намирането на единично решение, затова се изисква допълнителен механизъм, който позволява на ГА да намери друго решение (позиция на търсения еталон) след като едно (или повече) съвпадение е вече намерено. Този механизъм се заключава в следното: Когато се открие позиция в изходното изображение, в която стойността на NCC е по-голяма от 0.65 (стойността е определена емпирично), т. е. намерено е съвпадение с еталона, областта под еталона се променя с инверсно изображение съгласно (2).

$$I'(x,y) = 255 - I(x,y) \quad (2)$$

където I(x,y) и I′(x,y) са съответно старата и новата стойност на интензитета на пиксела в позиция с координати (x, y). По този начин при едно следващо съпоставяне с еталона в същата област ще се получи много малка стойност на взаимната корелация. Така съответната област се блокира за намиране на евентуални бъдещи решения. Алгоритъмът се прекратява, когато се открият всички търсени еталони или когато се достигне предварително указаният брой генерации (епохи) на ГА.

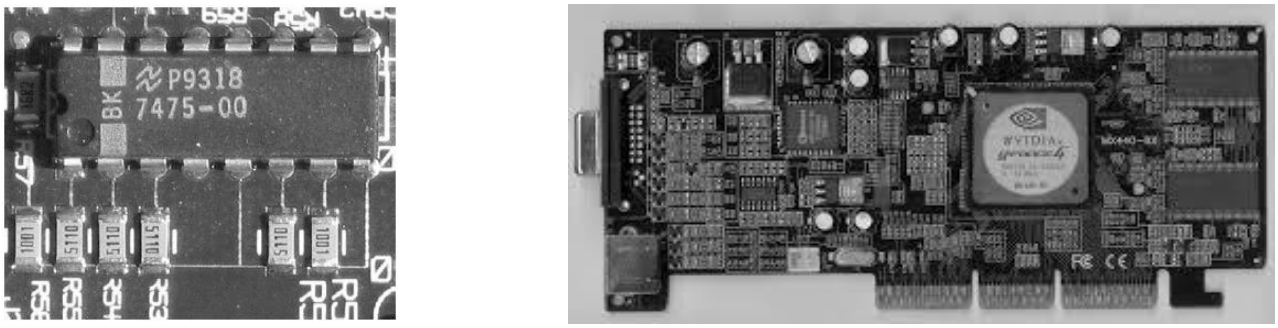
Блоковата схема на предложения ГА е представена на фигура 2.



Фиг. 2. Блокова схема на генетичния алгоритъм за локализиране на множество еталонни изображения

3. Експериментални резултати

За целите на експериментите са използвани два вида изображения на печатни платки, наситени с електронни елементи (фигура 3а и 3б).



а б

Фиг. 3. Наситени с елементи печатни платки за автоматична инспекция

Върху печатната платка от фигура 3а са разположени шест еднотипни резистора с правоъгълна форма. В случая те са разположени успоредно на ординатата, ако приемем, че началото на координатната система е в долния ляв ъгъл на печатната платка. Множеството от еталонни изображения, които трябва да бъдат идентифицирани върху печатната платка от фигура 3б, е набор от електролитни кондензатори, които са с кръгла форма.

И в двата случая се създава обобщен еталон, с който се извършват сравненията. Еталоните се създават чрез наслагване на отделните изображения на елементите, които ще се търсят и които присъстват върху печатните платки. С този обобщен еталон става възможно откриването на елементи, които са разположени под различен ъгъл.

С цел намаляване на изчислителната сложност на алгоритмите, се прилага и ограничаване на пространството на търсене, като за целта се използва предварителна обработка както на изображението, така и на еталона, целяща отделянето на подмножество от информативни точки (в случая точки от ръбовете на елементите на изображенията), по които да става изчисляването на ценовата функция.

Хромозомата, с която се извършва последователно търсене на еталона, е представена на фигура 4а. Нейните гени са: координата по оста Ox - x_i (отместване спрямо горния ляв ъгъл по хоризонталната ос), координата по оста Oy - y_i (отместване спрямо горния ляв ъгъл по вертикалната ос), ъгъл на завъртане и мащабен коефициент - α_i. Координатите са цели числа, а за кодиране на ъглите и мащабните коефициенти се използват по два бита.

x_i	y_i	α_i

Фиг. 4а. Хромозома за последователно търсене на еталонните изображения

Хромозомата, с която се търсят всички еталонни изображения едновременно (фигура 4б), се състои от толкова на брой (в случая с резисторите – шест) гена, колкото е предварително известният брой на търсените еталони. Всеки от гените е със структурата на единичния ген, използван при последователното търсене.

x_1	y_1	α_1	x_2	y_2	α_2	…	…	...	x_n	y_n	α_n

ген

Фиг. 4б. Хромозома за едновременно търсене на еталонните изображения

Ценовата функция при единичното търсене е нормализираната взаимна корелация. Нормализираната взаимна корелация в точката от изображението с координати *(x, y)*, изчислена за множество от представителни точки, е (3):

$$NCC(x,y)=\frac{\sum_{q=1}^{P} I(x+i_q,y+j_q).T(i_q,j_q)}{\sqrt{\sum_{q=1}^{P}\left(I(x+i_q,y+j_q)\right)^2}\cdot\sqrt{\sum_{q=1}^{P}\left(T(i_q,j_q)\right)^2}} \quad (3)$$

където $T(i_q,j_q)$ за q = 1,2,…,P е интензивността в q-тата избрана представителна точка от еталона, $I(x+i_q,y+j_q)$ е интензивността в точка от изображението с координати $(x+i_q,y+j_q)$. Множеството от представителните точки от еталона е $T=\{T_1(i_1,j_1),T_2(i_2,j_2),\ldots,T_P(i_P,j_P)\}$.

Ценовата функция, която е използвана при едновременното търсене, също е нормализираната взаимна корелация (3). Като още по-надеждна мярка би могло да бъде използван и корелационният коефициент (5), тъй като той намалява влиянието на осветеността на сравняваните изображения.

Корелационният коефициент върху избраните представителни точки от еталона в точката с координати (x,y) от изображението се изчислява съгласно формула (4):

$$NCCE(x,y)=\frac{\sum_{q=1}^{P}\left(I_q-\bar{I}\right).\left(T_q-\bar{T}\right)}{\sqrt{\sum_{q=1}^{P}\left(I_q-\bar{I}\right)^2}\cdot\sqrt{\sum_{q=1}^{P}\left(T_q-\bar{T}\right)^2}} \quad (4)$$

където:

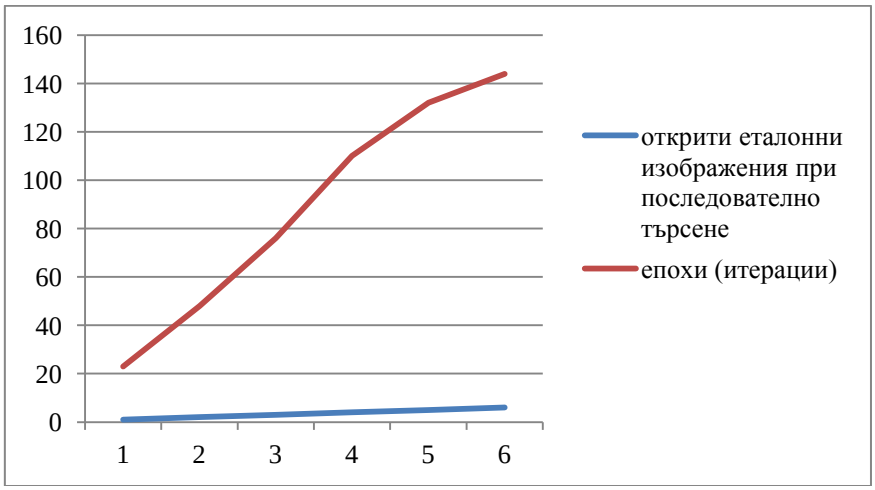
$$\bar{T}=\frac{\sum_{q=1}^{P}T_q}{P} \quad (5а)$$

е средната стойност на интензивностите на избраните представителни точки от еталона q = 1, 2, …, P и

$$\bar{I}=\frac{\sum_{q=1}^{P}I_q}{P} \quad (5б)$$

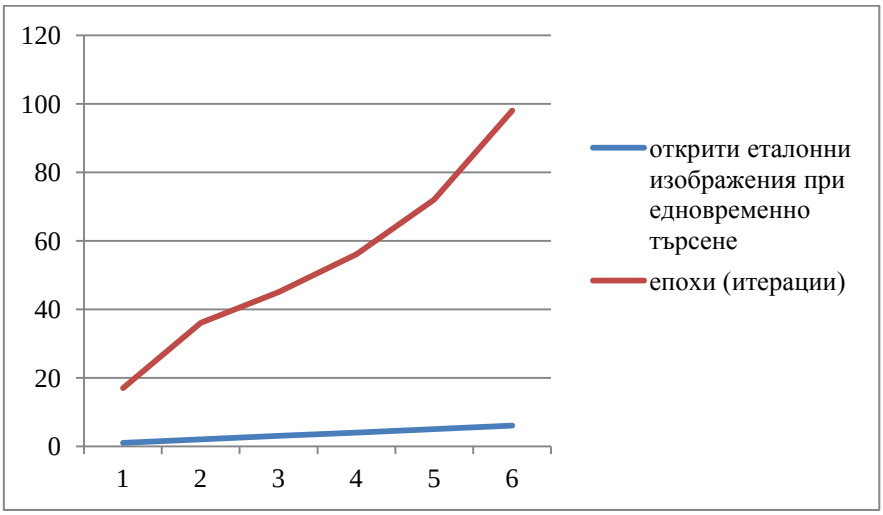
е средната стойност на интензивностите на съответстващите им по координати точки от изследвания прозорец от изображението.

На фигура 5 е показана зависимостта на броя на откритите еталонни изображения от броя на епохите (итерациите) на генетичния алгоритъм при последователно локализиране. Представени са усреднените резултатите от 100 реализации на алгоритъма.



Фиг. 5. Брой открити еталони във функция от итерациите на ГА при последователно търсене

На фигура 6 е показана зависимостта на броя на откритите еталонни изображения от броя на епохите (итерациите) на генетичния алгоритъм при едновременно локализиране на предварително известния брой еталони. Представени са усреднените резултатите от 100 реализации на алгоритъма.



Фиг. 6. Брой открити еталони във функция от итерациите на ГА при едновременно търсене

От получените резултати се вижда, че при едновременното търсене на еталонните изображения, чийто брой е предварително известен, резултатите се получават за по-малък брой итерации. Не може да се твърди същото за времето на изпълнение на алгоритъма, тъй като изчисленията при едновременното търсене са по-сложни и изискват повече време. Това се дължи на по-големия размер на хромозомата.

4. Заключение

Проблемът с регистрацията на изображения е много сложен проблем от областта на обработката на изображения. Приносът на тази статия се отнася ди използването на генетичен алгоритъм за намиране на съвпадение и регистриране на съответствие между две изображения, от които едното се смята за еталонно и то присъства в другото в един ии повече екземпляра. Получените експериментални резултати са обнадеждаващи, което означава, че предложеният подход може да намери успешно приложение в редица области и в частност – за автоматизирана инспекция на качеството на електронни изделия по отношение на броя и позициите на монтираните върху тях електронни елементи.

Литература

[1]. Bai Li, Li-Gang Gong, Ya Li, A Novel Artificial Bee Colony Algorithm Based on Internal-Feedback Strategy for Image Template Matching, The Scientific World Journal Volume 2014 (2014), Article ID 906861, 14 pages, http://dx.doi.org/10.1155/2014/906861

[2]. YIP Chi Lap, WONG Ka Yan, Efficient and effective tropical cyclone eye fix using genetic algorithms,
http://www.sc.ehu.es/ccwgrrom/KES2005/KES2004-IIPRS-papers/IIPRS-2/KES2004.pdf

[3]. Atul Pawar, Kajal Kamble, Shalaka Dhaktode, Dipali Dongare, Pooja Dushing, A Comparative Study of Emotion Recognition from Facial Expression, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 2, February 2015

[4]. Flávio Luiz Seixas, Luiz Satoru Ochi, Aura Conci, Débora C. M. Saade, Image Registration Using Genetic Algorithms, GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA, ACM 978-1-60558-131-6/08/07.
[5]. Nuske, S., Roberts, J. and Wyeth, G. Extending the Dynamic Range of Robotic Vision, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 2006, 162-167.
[6]. Silva, L., Bellon, O.R.P. and Boyer, K.L. Precision Range Image Registration Using a Robust Surface Interpenetration Measure and Enhanced Genetic Algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, v27 (5), 2005, 762-776.
[7]. Machowski, L.A. and Marwala, T. Evolutionary Optimisation Methods for Template Based Image Registration, School of Electrical and Information Engineering, 2004.
[8]. Crispin A. J., Rankov V., Automated inspection of PCB compomnents using genetic algorithm template matching approach, Int. J. Adv. Manuf. Technol, 2007, 35: pp. 293 – 300
[9]. A. Cenys, D. Gibavicius, N. Goranin, L. Marozas, Genetic Algorithm based Palm Recognition Method for Biometric Authentication Systems, Elektronika IR Elektrotechnika, ISSN 1392-1215, Vol. 19, No. 2, 2013, pp. 69-74
[10]. Карова Милена, Изследване и реализация на генетични алгоритми за решаване на един клас задачи, Дисертация за ОКС „Доктор", Варна, 2006 г.

За контакти:
гл. ас. д-р Юлка П. Петкова,
катедра „Компютърни науки и технологии",
Технически университет – Варна,
e-mail: yulka.petkova@tu-varna.bg

АНАЛИЗ НА СХЕМИТЕ ЗА ДЕГРАДАЦИЯ НА АРХИТЕКТУРИТЕ ЗА БЕЗОПАСНОСТ MooN

Петър Ц. Антонов

Резюме: Разглеждат се проблеми на функционирането на архитектурите за безопасност MooN. Предлага се подход за анализ на схемите за деградация на тези архитектури с използване на Марковските случайни процеси.
Ключови думи: безопасност, MooN, деградация, надеждност.

ANALYSIS OF DEGRADATION SCHEMES OF MooN SAFETY ARCHITECTURES

Peter Ts. Antonov

Abstract: The paper considers problems of the functioning of MooN safety architectures. An approach for analysis of degradation schemes of these architectures with using of Markov's random processes is offered.
Key words: safety, MooN, degradation, dependability.

Въведение

Както е известно, много технологични процеси и дейности са свързани с рискове за човешкия живот и околната среда. В тези случаи е наложително да се изградят и съответни системи, осигуряващи желаното ниво на безопасност. При разработването на такива системи следва да се отчита и връзката между безопасността, надеждността и сигурността и необходимостта от комплексен подход към решаването на тези проблеми [1]. На практика, желаното ниво на безопасност се достига с използването на схеми за защита, включващи различни механични, хидравлични, пневматични, електрически, електронни и програмируеми електронни компоненти. Електрическите и електронните компоненти се прилагат за тази цел в продължение вече на много години. Понастоящем с все по-нарастващи темпове за осигуряване на необходимото ниво на безопасност се използват и компютърните системи (наричани и програмируеми електронни системи - PES). Очевидно е, че за ефективната разработка и експлоатация на системите за безопаснст, основани на електрически и/или електронни компоненти и/или на компютърни системи е важно да бъдат приети и съответни международни и национални стандарти, както и препоръки за тяхното практическо приложение. Към настоящия момент основен в тази сфера е стандартът IEC 61508. Functional Safety of Electrical/Electronic/Programmable Electronic Safety - Related Systems [4].

Първата редакция на IEC 61508 е от 1998-2000 гг. и включва седем части: общи изисквания; изисквания към електрическите/електронните/програмируемите електронни системи (E/E/PES), отнасящи се към безопасността; изисквания към програмното осигуряване; определения и съкращения; примери и методи за определяне на нивата за безопасност; ръководство за прилагане на стандарта; обзор на методите и средствата [2]. Втората редакция е от 2010 г. и включва същите седем части, но с повишени изисквания към безопасността.

Стандартът IEC 61508 дефинира унифициран подход към въпросите за осигуряване на безопасността на системите, изградени на базата на електрически и/или електронни и/или

програмируеми електронни компоненти (E/E/PES). Той се явява като единна обща основа за разработване на конкретизирани стандарти за различните предметни области и на препоръки за практическо използване.

С практическото внедряване на IEC 61508 започна да се налага препоръчаната в него тенденция за преминаване от качествен към количествен анализ на безопасността. Стандартът, обаче, не предлага конкретни методи и предписания за осъществяване на такъв анализ. Това обосновава необходимостта от допълнителни изследвания и разработки в това направление. В тази връзка, в настоящата работа се разглеждат проблеми на функционирането на архитектурите за безопасност MooN и се предлага подход за анализ на схемите за деградация на тези архитектури с използване на апарата на Марковските случайни процеси.

Математически модели за анализ на схемите за деградация на архитектурите MooN

Архитектурите MooN (M out of N) са определени за пръв път в стандарта IEC 61508 като архитектури за осигуряване на необходимата безопасност на технологични процеси и дейности [3,4]. В тези архитектури се използва един от основните методи за осигуряване на необходимите надеждност и безопасност в съвременните условия - метода на резервирането. Резервирането в случая се реализира като постоянно резервиране с гласуване. Както е известно, отличителният признак на резервираните системи с гласуване е невъзможността да се отделят основните компоненти от резервните, тъй като всички компоненти работят едновременно. Означението MooN е обобщено описание на множество архитектурни схеми като 1oo1, 1oo1D (наличието на буквата D означава, че в дадената архитектура има вградена подсистема за диагностика на неизправностите), 1oo2, 1oo2D, 2oo2, 2003 и др.

В процеса на функциониране на MooN са възможни откази, в случайни моменти от времето, на техни компоненти, които водят след себе си до преходи (преминаване) от една архитектура към друга, с възможност за възстановяване. Последователностите на тези преходи се наричат схеми за деградация [3 и др.]. Аналитичното описание на процеса на функциониране на такива схеми е възможно да се направи с апарата на Марковските случайни процеси с дискретни състояния и непрекъснато време.

За графичното представяне на Марковски случаен процес с дискретни състояния и непрекъснато време може да се използва ориентиран граф, върховете на който представляват отделните състояния, а всяка свързваща два върха дъга - прехода от едно състояние в друго, което се осъществява в произволни моменти от времето. На базата на такъв граф на състоянията и преходите могат да се съставят диференциални уравнения на състоянията, като за целта се разглежда всяко едно състояние в даден момент от времето t и се прогнозират вероятните следващи състояния, след достатъчно малък интервал от време Δt ($\Delta t \rightarrow 0$). При това се използва следното лесно доказуемо твърдение: ако Марковски случаен процес с дискретни състояния и непрекъснато време се намира в състояние S_i и под въздействието на случаен Поасонов поток с интензивност λ_{ij} преминава в състояние S_j за интервал от време Δt ($\Delta t \rightarrow 0$), то вероятността за този преход може да се приеме за равна на $\lambda_{ij}\Delta t$. Решението на получената система от диференциални уравнения, което ще представлява измененията във времето на вероятностите за пребиваване в отделните състояния, позволява да се направи анализ на цялостното поведение на конкретната изследвана система.

По-нататък в настоящето изложение се разглеждат схемите за деградация на най-простите и често използвани резервирани архитектури за безопасност MooN - 1oo2, 2oo2 и 2oo3 (1oo1 е нерезервирана архитектура).

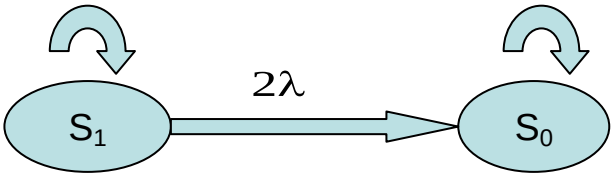
Нека λ - интензивност на възникване на отказите на компонентите в архитектурите MooN, а μ - интензивност на възстановяване на компонентите след отказ.

Модел на схемата за деградация на архитектурите за безопасност 1oo2 и 2oo2

Очевидно е, че при отказ на един компонент и двете архитектури ще деградират до 1oo1. Използването, обаче, на тази нерезервирана архитектура в много случаи може да се окаже опасно, поради което е целесъобразно тя да се разглежда като неработоспособна и схемата за деградация в случая да се запише по следния начин: (2oo2, 1oo2) - (1oo1, 0) (с нула е обозначено състоянието, в което архитектурата престава да функционира).

В [3] е приведен много удачен пример за опасност на архитектурата 1oo1, до която се деградира след отказ на един компонент на 2oo2. Ако в подводна лодка за контрол на херметичността на люка е изградена система 2oo2 с два датчика, то за осигуряване на безопасността сигналът за потапяне трябва да се получи само при потвърждение и от двата датчика едновременно. При положение, че един от датчиците откаже, тогава архитектурата ще деградира до 1oo1. Ако тази архитектура се остави да функционира и единият датчик сработи лъжливо, то ще последва потапяне на лодката при отворен люк. Можем да посочим и редица други случаи, например, използване на архитектурата 2oo2 в системите за пожарна сигнализация. При отказ на един от датчиците, другият датчик в 1oo1 може да сработи лъжливо. Този случай по принцип не е опасен, но е свързан с материални разходи, поради ненужното задействане на средствата за пожарогасене. Очевидно е, че много по-сериозна ще бъде ситуацията, ако възникне пожар и датчикът не среагира. Затова деградацията до нерезервираната архитектура 1oo1 е разумно да се разглежда като равностойна на деградацията до състояние 0.

Графът на състоянията и преходите на схемата за деградация (2oo2, 1oo2) - (1oo1, 0) е представен на фиг. 1, където S_1 е състояние на работоспособност, а S_0 - състояние на неработоспособност, т.е. състояние, в което архитектурата престава да функционира.



Фиг. 1.

Системата от диференциални уравнения за този прост граф, където $P_{S1}(t)$ и $P_{S0}(t)$ са вероятностите за пребиваване в състояние S_1 и съответно S_0, се получава по следния начин:

$$P_{S1}(t+\Delta t) = P_{S1}(t)(1-2\lambda\Delta t)$$
$$P_{S0}(t+\Delta t) = P_{S1}(t)2\lambda\Delta t + P_{S0}(t).$$

След разкриването на скобите в първото съотношение, прехвърляне на съответните вероятности в левите части на двете съотношения и разделяне на двете страни на съотношенията с Δt ($\Delta t \to 0$), можем да запишем окончателно

$$\frac{dP_{S1}(t)}{dt} = -2\lambda P_{S1}(t)$$
$$\frac{dP_{S0}(t)}{dt} = 2\lambda P_{S1}(t).$$

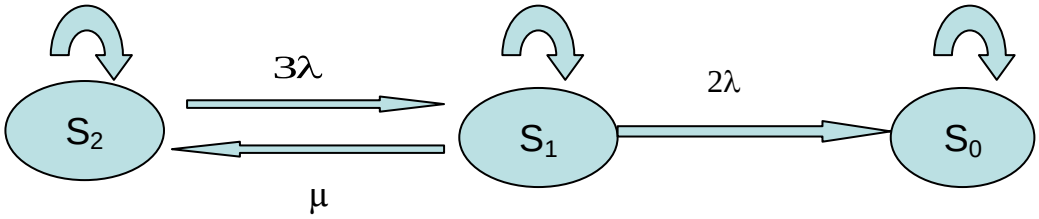
Решението на тази система от две диференциални уравнения ще представлява вероятностите, във функция от времето t, за работоспособност - $P_{S1}(t)$ и съответно неработоспособност - $P_{S0}(t)$, на архитектурите 1oo2 и 2oo2.

Модел на схемата за деградация на архитектурата 2oo3

При отказ на един от компонентите на резервираната архитектура 2oo3, нейното ниво на безопасност се понижава, но тя може да продължи да функционира като архитектура 2oo2 или 1oo2. Затова, в общия случай за архитектурата 2oo3 могат да бъдат посочени следните два варианта на схемата за деградация:

2oo3 - 2oo2 - 1oo1 - 0 и
2oo3 - 1oo2 - 1oo1 - 0,

които могат да бъдат представени съвместно, като 2oo3 - (2oo2, 1002) - 1oo1 - 0. Имайки в предвид, обаче, казаното по-горе за опасността на нерезервираната архитектура 1oo1, схемата за деградация на 2oo3 е по-целесъобразно да се запише по следния начин: 2oo3 - (2oo2, 1oo2) - (1oo1, 0). Графът на състоянията и преходите за тази схема на деградация е представен на фиг. 2, където S_2 е състояние на работоспособност и на трите компонента, т.е. 2oo3, S_1 - състояние на деградация до 2oo2 или 1oo2, а S_0 - състояние на неработоспособност.



Фиг. 2.

Системата от диференциални уравнения за този граф се получава по следния начин:

$$P_{S2}(t+\Delta t) = P_{S1}(t)\mu\Delta t + P_{S2}(t)(1-3\lambda\Delta t)$$
$$P_{S1}(t+\Delta t) = P_{S2}(t)3\lambda\Delta t + P_{S1}(t)(1-\mu\Delta t - 2\lambda\Delta t)$$
$$P_{S0}(t+\Delta t) = P_{S1}(t)2\lambda\Delta t + P_{S0}(t)$$

и окончателно

$$\frac{dP_{S2}(t)}{dt} = -3\lambda P_{S2}(t) + \mu P_{S1}(t)$$
$$\frac{dP_{S1}(t)}{dt} = -(\mu + 2\lambda)P_{S1}(t) + 3\lambda P_{S2}(t)$$
$$\frac{dP_{S0}(t)}{dt} = 2\lambda P_{S1}(t).$$

В случая работоспособните състояния са S_2 и S_1, поради което финалната вероятност за работоспособност на архитектурата 2oo3 ще се получи като сума от вероятностите $P_{S2}(t)$ и $P_{S1}(t)$.

Следва още да се отбележи, че в някои случаи нерезервираната архитектура 1oo1 може да се разглежда временно и като работоспособна. Това предполага и друга възможна схема за деградация на 1oo2, а именно: 1oo2 - 1oo1 - 0. За този случай графът на състоянията и преходите ще изглежда като на фиг. 2, но със следните разлики: състояние S_2 ще съответства на 1oo2, състояние S_1 - на 1oo1, а S_0 - на 0; интензивността на прехода $S_2 \rightarrow S_1$ ще бъде 2λ, а на прехода $S_1 \rightarrow S_0$ - само λ.

Заключение

В заключение ще отбележим, че изложеният по-горе подход за анализ на схемите за деградация на 1oo2, 2oo2 и 2oo3 може да се използва и при всяка друга архитектура за безопасност MooN. Това се отнася и за архитектурите с подсистеми за диагностика MooND. За решаването на системите от диференциални уравнения могат да се използват редица програмни продукти като MATLAB и др.

Освен това, както бе отбелязано по-горе, проблемите на безопасността следва да се разглеждат съвместно с проблемите на надеждността и сигурността, тъй като недостатъчно надеждните и сигурни системи за безопасност могат да се превърнат в особено опасни. Самите системи за безопасност обикновено се изграждат като комплексни (интегрирани) системи, включващи подсистеми с различно предназначение: за контрол и управление на достъпа, за пожароизвестяване, видеонаблюдение и др.

Литература

[1]. Антонов, П. Ц. Таксономични и терминологични проблеми в областта на надеждността, безопасността и сигурността и подход за комплексен анализ. В: Сб. научни трудове на между-народна конференция'2008. Шумен, ШУ, Унив. изд-во "Еп. К. Преславски", 2009, с. 142-151.

[2]. Скляр, В.В. Сертификация информационно-управляющей платформы на базе ПЛИС на соответствие требованиям по функциональной безопасности стандарта МЕК 61508. ISSN 2073-6237. Ядерна та радіаційна безпека, 4(60), 2013, с. 54-60.

[3]. Денисенко, В. Аппаратное резервирование в промышленной автоматизации. СТА, 2/2008, с. 90-92. (www.cta.ru).

[4]. IEC 61508 Overview Report. A Summary of the IEC 61508 Standard for Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. Exida, Sellersville, PA 18960, USA, 2006, 29 p.

За контакти:
доц. д-р инж. Петър Ц. Антонов
катедра КНТ,
ТУ-Варна
E-mail: peter.antonov@ieee.org;
antonovp@ieee.bg;
peter.antonov@tu-varna.bg

ПОДХОД ЗА ПЛАНИРАНЕ И ИЗПЪЛНЕНИЕ НА ПАРАЛЕЛНИ ЗАДАЧИ В РАЗПРЕДЕЛЕНА СРЕДА
(резюме на дисертация за получаване на образователна и научна степен "доктор")

Ивайло П. Пенев

Резюме: Статията представя резюме на автореферат към дисертационен труд. Обект на разглеждане са задачи с голям брой независими изчисления с големи обеми от данни. Предложен е алгоритъм за разделяне на задачите на подзадачи, подходящи за паралелно изпълнение от свързани изчислителните възли. Предложени са два евристични алгоритъма за планиране на паралелно изпълнение на задачите в изчислителните възли. Целта е намаляване на общото време за приключване на всички задачи в сравнение с последователното им изпълнение. Описаните алгоритми са приложени върху важни практически задачи: изчисляване на риск по метод Монте Карло и оценяване на фондове. Представени са резултати от експериментални изследвания.
Ключови думи: паралелизъм, планиране, евристични алгоритми, сортиране, генетичен алгоритъм, Монте Карло.

Approach for scheduling and execution of parallel jobs in a distributed computing environment
(summary of a dissertation for a doctor's degree)

Ivaylo P. Penev

Abstract: The paper presents a summary of the abstract of the PhD thesis. Problems with lots of independent calculations over huge amounts of data are explored. An algorithm for decomposition of the problems to jobs, suitable for parallel execution by connected workstations is proposed. Two heuristic algorithms for scheduling the execution of the tasks on the workstations are proposed. The target is reduction of the total time for termination of the jobs, compared to their sequential execution. The described algorithms are demonstrated for solving important problems: risk calculation by Monte Carlo method and estimation of funds. Results from the experiments are presented.
Keywords: parallelism, scheduling, heuristic algorithms, sorting, genetic algorithm, Monte Carlo.

1. Увод

Необходимостта от паралелни изчисления възниква в много научни и приложни области. Настоящият труд е насочен към изчислителни задачи, които могат да бъдат разделени на независими части, всяка от които се изпълнява от отделен процесор. Примери за такива задачи са моделиране на различни процеси чрез Монте Карло метод, оценяване на финансов риск, метеорологични прогнози, диагностициране на заболявания и др.

Паралелното изпълнение на задачите изисква подходяща изчислителна среда. Специализираните машини за високо производителни изчисления струват скъпо и се срещат рядко предимно в университети и изследователски институции. Всички организации разполагат с работни станции, чиято изчислителна мощност не се използва пълноценно. Създадени са технологии за организиране на изчислителни клъстери, които осигуряват средства за решаване на сложни изчислителни задачи чрез налични ресурси.

Реализацията на паралелни изчисления в разпределена среда е свързана с различни проблеми, които са свързани основно с комуникационната среда между изчислителните

възли. Ключови моменти за ефективността на паралелното изпълнение са разделянето на задачата на подзадачи и намаляване на комуникациите между подзадачите по време на паралелното изпълнение. Друг проблем е предотвратяване на конкурентния достъп на задачите до общ източник на данни, който се използва в много случаи, когато за паралелните изчисления са необходими множество натрупани статистически данни.

Работата в дисертационния труд е насочена към два основни проблема при паралелно изпълнение на голям брой задачи:

- Намаляване на предаваните съобщения между задачите по време на паралелното изпълнение.
- Предотвратяване на конкурентния достъп на задачите до общ източник на данни.

1.1. Формиране на паралелни задачи и проблеми при организацията на паралелни изчисления

В дисертационния труд се разглеждат задачи, които могат да бъдат разделени на независими подзадачи. Всяка подзадача може да бъде изпълнена от отделен процесор и всички подзадачи могат да бъдат изпълнявани едновременно. Разделянето на такива задачи не изисква специални техники или алгоритми. Всички подзадачи се изпълняват с различни (или еднакви) входни данни и за получаването на резултат не е необходимо никоя подзадача да използва резултати от други подзадачи. Освен начално разпределяне на входните данни е необходимо също събиране и комбиниране на резултати.

Типични примери за задачи, при решаване на които се използват паралелни изчисления, са:

- Задачи за изчисляване по Монте Карло

Монте Карло е метод за моделиране на процеси чрез използване на серии от случайни числени стойности с определени статистически свойства като прието разпределение и корелация. Използва се в случаите, когато построяването на математически модел за даден изследван процес е невъзможно или прекалено сложно. В основата на метода стоят изчислителни операции със случайни числа. Всяко изчисление е независимо от останалите, което прави Монте Карло особено подходящ за паралелна реализация.

Основна трудност при паралелната реализация е свързана с предаване на параметрите за изчисляване на оценяващата функция към компютрите в средата. Allen и Wilkinson предлагат подход, при който се използва отделен процес за генериране на следващо случайно число [6]. Първоначално главен (master) процес стартира подчинени (slaves) процеси, които получават числа от процеса генератор за всяко изчисляване на функцията. Подчинените процеси изчисляват своите частни суми и ги изпращат на главния процес, който събира получените резултати. В тази реализация главният процес изчаква заявка от всеки подчинен процес за изпращане на следващо случайно число. Това води до наличие на допълнителни комуникации, които намаляват ефективността на паралелното изпълнение.

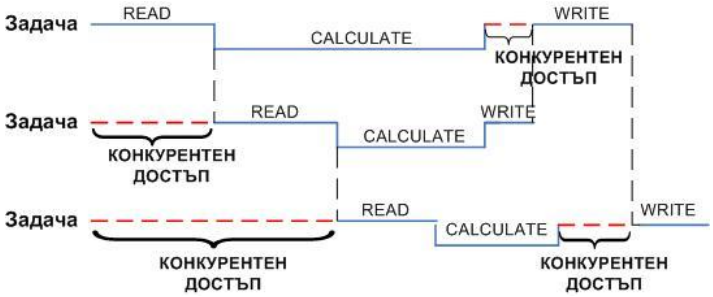
- Оценяване на фондове

Системи за управление на големи фондове от портфейли са практически пример за паралелни изчисления. Такива системи се използват във всяка съвременна финансова институция. Оценяването на портфейл се състои в изпълнение на предварително дефинирани симулационни анализи върху данни за инструментите на всеки портфейл. Крайният резултат са различни оценки за риска на инвестициите във фондовете.

Данните, върху които се изпълняват симулационните анализи, се съхраняват в общ източник (база от данни). В същия източник се записват и резултатите, получени от симулационните анализи.

Всяка задача се състои от три фази: четене на данни (READ), изчисления (CALCULATE) и запис на резултати (WRITE). При едновременно стартиране на задачите се получава конкурентен достъп до общия източник при четене на данни и при запис на

резултати. Едновременният достъп намалява ефективността на паралелното изпълнение (фиг. 1).



Фиг. 1. Влияние на конкурентния достъп при паралелното изпълнение

За преодоляване на конкурентния достъп на задачите до общия източник на данни при паралелното изпълнение могат да бъдат използвани следните подходи:

- Паралелно програмиране с обмен на съобщения

Стандартът MPI (Message Passing Interface) е най-популярната спецификация за обмен на съобщения, поддържаща паралелното програмиране. Той предоставя свободно достъпни библиотеки, които осигуряват преносимост при различни платформи с различна производителност. Основен недостатък на този подход е необходимостта от сериозни модификации на програмния код на съществуващите приложения.

- Чрез подходящо планиране на задачите за изпълнение

В теорията на планиране са известни методи и алгоритми за планиране на изпълнението на различни класове задачи при различни целеви функции и ограничителни условия.

1.2. Планиране на задачи

Най-известните методи и алгоритми за планиране на задачи при наличие на ограничителни условия са:

- Методи и алгоритми за намиране на оптимално решение

Към тази група принадлежат метод на критичния път (critical path method) [1], линейно и целочислено програмиране (linear and integer programming) [2], метод на разклоненията и границите (branch and bound method) [3]. Най-сериозният недостатък на тези методи е, че са неприложими при планиране на голям брой задачи.

- Евристични методи и правила

Евристичните методи намират оптимално решение в някои случаи на задачата за планиране, а най-често намират решение, което е близко до оптималното.

Най-често използваните евристични правила при планиране на различни класове задачи са:

- Подреждане на задачи в списък (List Scheduling);
- Приоритетно обработване на задача с най-голямо време за изпълнение (Longest Processing Time First);
- Приоритетно обработване на задача с най-малко време за изпълнение (Shortest Processing Time First).

При решаване на много практически задачи за планиране, особено при наличие на разнообразни ограничителни условия, се налага използване на по-сложни евристични правила чрез комбиниране на известните прости правила. Редица автори предлагат правила за планиране на задачи при различни ограничителни условия, напр. [4].

- Приложения на генетични алгоритми в планирането

Генетичните алгоритми са най-често използваните евристични алгоритми в задачите за планиране. Основните предимства, които ги правят подходящи за решаване на задачи за планиране са:

- възможност за лесно представяне на много различни ограничителни условия;
- търсене на решения в множество от точки в пространството от решения, с което се избягва попадане в локални екстремуми на целевата функция.

Известни са генетични алгоритми за планиране на задачи (напр. алгоритъм на Wall [5]). При наличие на ограничени ресурси в средата получените резултати са незадоволителни (с големи отклонения от съответните оптимални решения). Подобряването на ефективността на алгоритъма за планиране на задачи може да се постигне чрез съобразяване на кодирането на кандидат-решенията със специфичните особености на разглеждания клас задачи за планиране.

2. Алгоритми за планиране и изпълнение на паралелни задачи

Предложеният подход се състои от следните стъпки:

1. Разделяне на задача на независими задачи.
2. Разпределяне на задачите между изчислителните възли (компютрите) в средата.
3. Планиране на задачите за паралелно изпълнение в дадена разпределена.
4. Стартиране на задачите според получения план в разпределената среда.

2.1. Формиране и изпълнение на паралелни задачи

Предложен е паралелен алгоритъм, при който са редуцирани предаваните съобщения между задачите по време на изпълнение. Алгоритъмът използва разделяне на данни между slave процесите и премахва допълнителните съобщения за получаване на случайни числа от всеки slave процес.

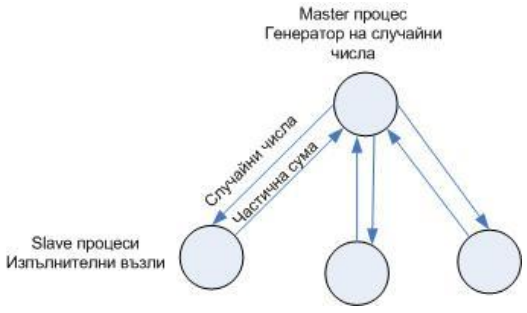
Предложеният алгоритъм е демонстриран за паралелно изчисляване на риск на финансов инструмент по Монте Карло метод в среда от m еднакви изчислителни възли с n случайни числа.

Стъпки на алгоритъма:

1. Master процес изпраща n/m числа на всеки от изчислителните възли (slave процеси).
2. Възлите изчисляват оценяваща функция едновременно.
3. Възлите изпращат получените резултати на master процеса.
4. Master процесът събира получените резултати и конструира краен резултат.

Предложеният алгоритъм има две основни разлики в сравнение с известни паралелни алгоритми:

- генерирането на случайни числа е реализирано в master процеса, т.е. master процесът е обединен с процеса генератор на случайни числа (фиг. 2);



Фиг. 2. Паралелен алгоритъм за изчисляване по Монте Карло метод

- премахнати са съобщенията от slave процесите за получаване на случайни числа, т.е. елиминиран е режимът на ръкостискане.

2.2. Планиране на изпълнението на паралелни задачи при ограничени ресурси

В дисертационния труд е предложен подход с отлагане на стартирането на задачи, при който се използват предварително оценени времена за изпълнение на задачите. Целта е да се предотврати конкурентният достъп до общия източник на данни. Симулират се сценарии с отместване на задачите една спрямо друга. Целта е да се намери такъв план на изпълнение, при който във всеки времеви интервал само една задача да има достъп до общия източник на данни.

Задачата за планиране на n задачи в среда от m свързани компютри е формулирана като оптимизационна задача. *Целева функция* е времето за приключване на всички задачи в плана. *Времевите ограничения* определят ред на изпълнение на подзадачите за всяка задача: четене на данни, изчисления, запис на резултати. *Ресурсните ограничения* определят отсъствие на конкурентен достъп на задачите до общия ресурс. *Приемливо решение* на задачата за планиране е намиране на стартов момент за всяка задача, при което са удовлетворени поставените времеви и ресурсни ограничения. *Оптимално решение* е минималната от всички възможни стойности на времето за приключване на последната задача в плана, т.е. минималната стойност на целевата функция при удовлетворени ограничения.

Описаната задача за планиране е сведена до задача за минимално оцветяване на върховете на граф, което доказва нейната NP-трудност [7]. Класифицирането на задачата като NP-трудна означава, че за решаването и в общия случай следва да се използват евристични алгоритми.

В дисертационния труд са предложени алгоритми, решаващи задачата за планиране в два случая:

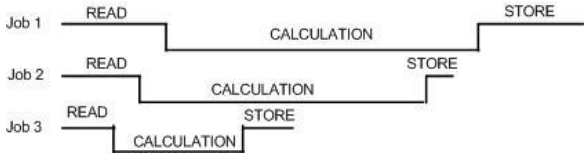
1. Задачите са не повече от компютрите в средата, т.е. при $n \leq m$.
2. Задачите са повече от компютрите в средата, т.е. при $n > m$.

2.3. Алгоритъм със сортиране на задачите при $n \leq m$ (задачите са не повече от компютрите)

Идеята на алгоритъма е приоритетно стартиране на задача, за която са приключени четенето на данни и изчисления. Такава задача е готова за запис на данни в източника и за приключване на изпълнението си.

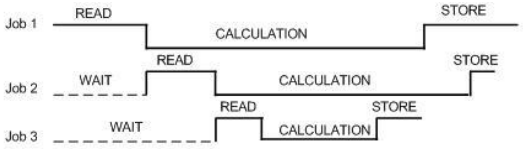
Стъпки на алгоритъма:

Стъпка 1. Сортиране на задачи по време за изпълнение в низходящ ред (фиг. 3)



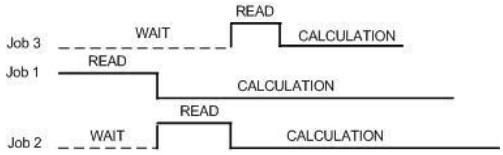
Фиг. 3. Първо сортиране на списъка от задачи

Стъпка 2. Отместване на задачи по оста на времето с цел предотвратяване на конкурентен достъп при четене на данни (фиг. 4)



Фиг. 4. Отместване на задачи

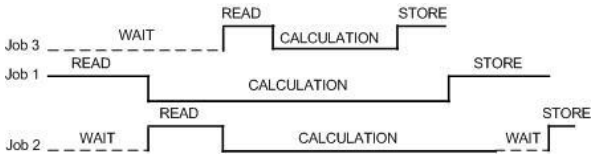
Стъпка 3. Сортиране на задачи по време за четене и време за изчисления във възходящ ред (фиг. 5)



Фиг. 5. Второ сортиране на списъка от задачи

В този случай задачата Job_3 първа е приключила подзадачите четене на данни и изчислителни операции. Следователно тази задача е готова за съхраняване на резултати. Поради това тази задача трябва да бъде стартирана първа в плана.

Стъпка 4. Планиране на подзадача за съхраняване на резултати (операция STORE) за всяка задача и получаване на окончателен план за изпълнение на всички задачи (фиг. 6)



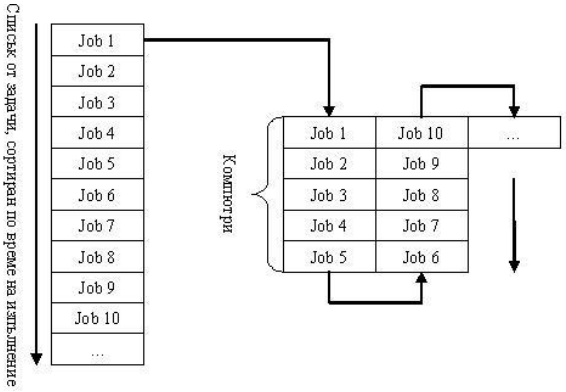
Фиг. 6. Окончателен план за изпълнение на задачите

2.4. Решаване на задачата за планиране при n>m (задачите са повече от компютрите)

2.4.1. Разпределяне на задачи между компютрите

В дисертационния труд е реализиран алгоритъм за разпределяне на задачите между компютрите, чиято цел е да осигури приблизително равно време за работа на всички компютри в средата.

Задачите се сортират по времена за изпълнение в низходящ ред. След това първите *p* задачи от формирания списък се присвояват съответно на първите *p* компютъра. Задачите с индекс от *p+1* до *2p* се разпределят в обратен ред, съответно към компютри от *p+1* до 1. Тези две стъпки се редуват докато съществуват неразпределени задачи (фиг. 7).



Фиг. 7. Алгоритъм за разпределяне на задачите

2.4.2. Генетичен алгоритъм за намиране на план

Разработен е генетичен алгоритъм, който избира последователно най-добрите кандидати сред множество (популация) от решения. За целта за всяко кандидат-решение се изчислява целева функция, определяща брой времеви интервали, в които две или повече задачи достъпват едновременно общия източник на данни. На всяка итерация алгоритъмът генерира изместване при стартирането на произволно избрана задача. Ако отложеното стартиране на задачата удовлетворява поставеното ограничително условие и постига по-добра стойност на целевата функция от предишни решения (т.е. план с по-малка продължителност), то полученото решение се запомня като по-добро от предходните решения.

• Кодиране на хромозомите

Хромозомата представлява списък с всички изчислителни възли (компютри) в средата. Всеки елемент от списъка представлява една задача или последователност от задачи, изпълнявани от компютър, със съответните оценени времена за отложено стартиране, четене на данни, изчисления и запис на резултати в общия източник.

Всяка хромозома е представена като символен низ, чиито елементи имат една от две възможни стойности – 0 или 1. Времевите интервали, в които съответната задача използва общия източник, са кодирани със стойност „1”, а останалите със стойност „0” (фиг. 8).

Фиг. 8. Структура на низ, представящ хромозома

• Целева функция

Целевата функция в генетичния алгоритъм е линейна комбинация от наказателна функция и оценъчна функция:

$f(g(p), h(C_{max}))$,

където:

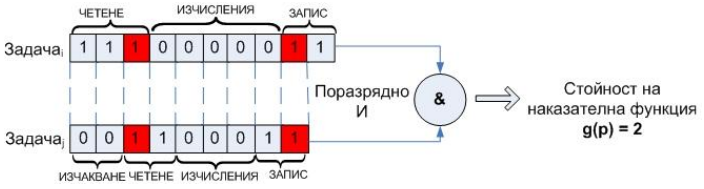
f - целева функция, оценяваща качеството на получено решение (план за изпълнение на всички задачи),

$g(p)$ - наказателна функция, оценена чрез брой интервали, в които две или повече задачи извършват конкурентен достъп до общия ресурс,

$h(C_{max})$ - функция, представяща времето за приключване на всички задачи (общата продължителност на плана).

Дадено решение е приемливо, ако $g(p) = 0$, т.е. във всеки момент от времето само една задача използва общия източник на данни.

Наказателната функция се изчислява чрез сечение на низовете на всички двойки компютри от текущата хромозома (фиг. 9). Броят на стойностите 1 в получения резултантен низ се прибавя към сумата на времевите интервали с конкурентен достъп до общия ресурс в генерирания план.



Фиг. 9. Изчисляване на наказателна функция

• Избор на условие за прекратяване на еволюционния процес

В генетичния алгоритъм са използвани две условия за прекратяване на еволюционния процес.

1. Достигане на максимален брой популации

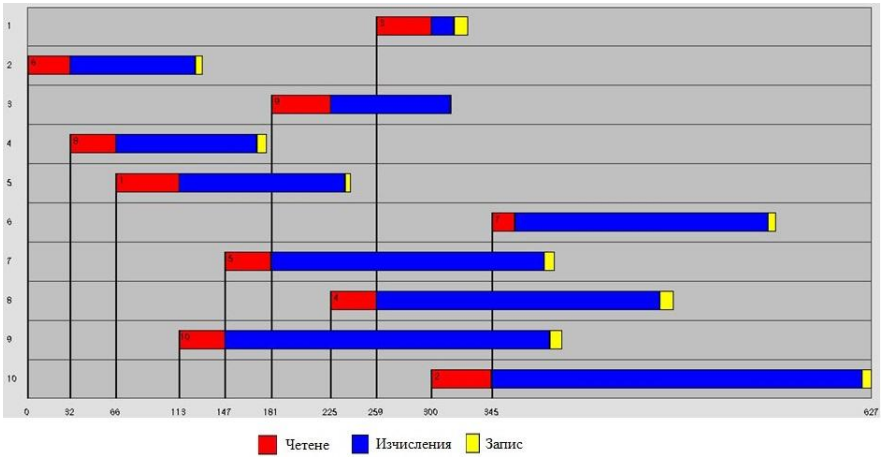
Алгоритъмът е изследван с максимален брой популации, вариращ от 100 до 10000.

2. Стабилизиране на популацията

Генерирането на нови популации продължава докато в новата популация стойността на целевата функция престане да се променя.

• Генериране на начална популация

Началната популация се създава чрез случайно избиране, изместване на задачи и изчисляване на целевата функция за всички хромозоми от популацията (фиг. 10).



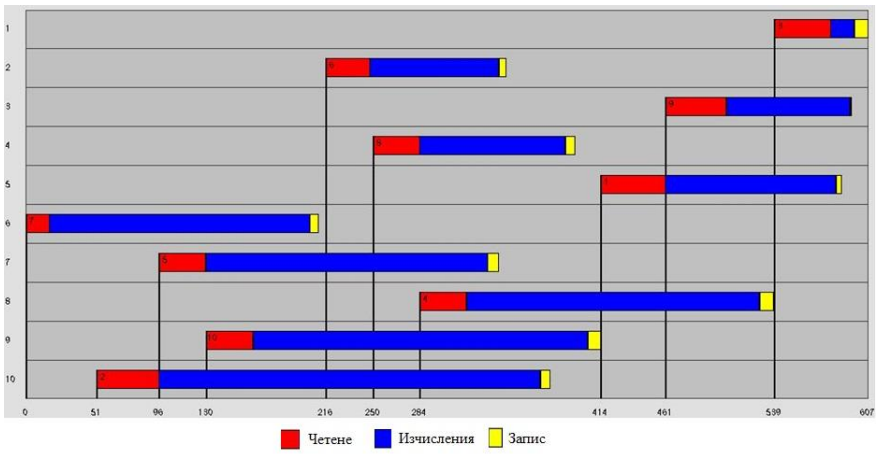
Фиг. 10. Начална популация с време за приключване на всички задачи 627 времеви интервала

- Селекция

Всяка следваща популация се създава като копие на предишната, но към хромозомите на новата се прилага оператор мутация.

- Мутация

Мутацията се изпълнява за произволен компютър от хромозомата, като се избира произволна задача от него. Определя се дали изместването на задачата по оста на времето може да намали общия брой на времевите интервали с конкурентен достъп до общия източник в хромозомата. Това става чрез изчисляване на целевата функция само за избрания компютър. Ако изместването на задачата води до получаване на хромозома с по-добра стойност на целевата функция от всички получени до момента, то задачата се измества и изменената чрез мутацията хромозома се запомня в популацията (фиг. 11).



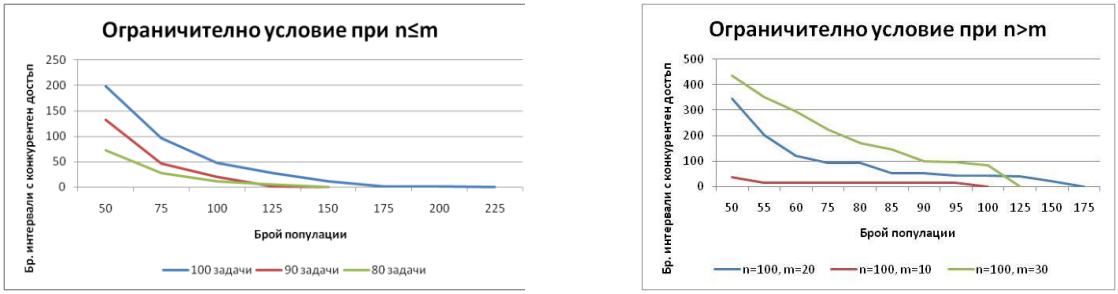
Фиг. 11. Популация, получена след прилагане на мутация (време за приключване на всички задачи – 607 времеви интервала)

3. Експериментални изследвания и резултати

3.1. Експериментално изследване на генетичния алгоритъм

Изследвано е влиянието на броя на популациите в генетичния алгоритъм върху наказателната част от целевата функция, т.е. върху броя на интервалите с конкурентен достъп в полученото решение (изпълнението на ограничителното условие).

Генетичният алгоритъм е изпълнен с различен брой задачи. За всеки брой задачи алгоритъмът е тестван с различен брой популации. Измерен е броят на интервалите с конкурентен достъп в полученото решение (фиг. 12).



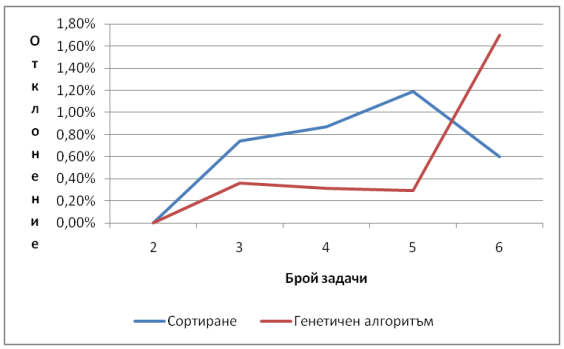
Фиг. 12. Влияние на броя на популациите върху ограничителното условие

Изводът от направения експеримент е, че сравнително малък брой популации на генетичния алгоритъм удовлетворяват ограничителното условие дори и при голям брой задачи и компютри. Около 200 популации са достатъчни за предотвратяване на конкурентния достъп при планиране на 100 задачи.

3.2. Изследване на получените решения от съответните оптимални при предложените евристични алгоритми

3.2.1. При n≤m (задачите са не повече от компютрите в средата)

Изследвано е отклонението на продължителността на плана за двата евристични алгоритъма от продължителността на оптималния план, получен чрез изчерпващия алгоритъм (фиг. 13).

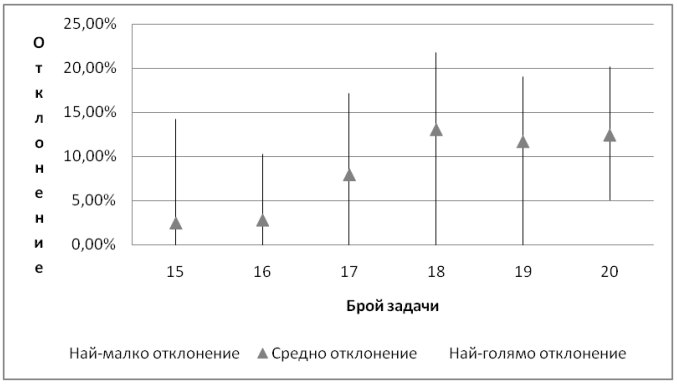


Фиг. 13. Отклонение на решенията на евристичните алгоритми от оптимално решение, получено чрез изчерпващ алгоритъм при n≤m

И при двата алгоритъма регистрираното отклонение от продължителността на оптималния план е минимално (под 2%). При генетичния алгоритъм се наблюдава влошаване на качеството на получените решения с увеличаване на броя на задачите. Следователно алгоритъмът със сортиране е по-подходящ за планиране на задачи в случая, когато наличните компютри в средата са не по-малко от задачите за изпълнение (n≤m).

3.2.2. При n>m (задачите са повече от компютрите в средата)

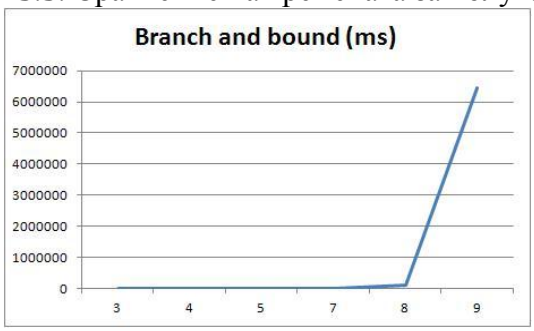
Генетичният алгоритъм е изпълнен за генериране на план с различен брой задачи при различен брой компютри в средата. Измерени са най-голямо, най-малко и средно отклонение на времето за приключване на плана от съответното оптимално време, получено чрез изчерпващ алгоритъм (фиг. 14).



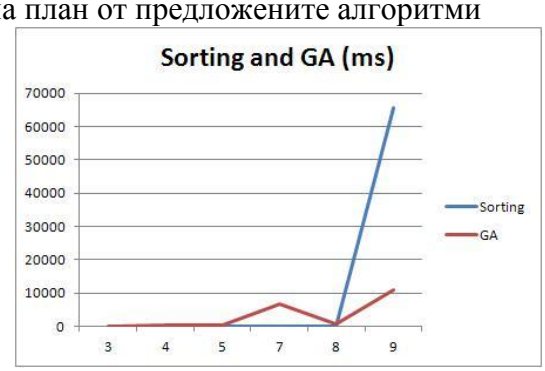
Фиг. 14. Отклонения от оптималното решение при 10 компютъра

Резултатите показват, че представените евристични алгоритми генерират планове на изпълнение, чиято продължителност се отклонява от продължителността на съответния оптимален план значително по-малко в сравнение с отклоненията при планове, получени чрез известни генетични алгоритми за планиране при ограничени ресурси.

3.3. Сравнение на времената за получаване на план от предложените алгоритми



а) Време за получаване на план от алгоритъма с пълно изчерпване



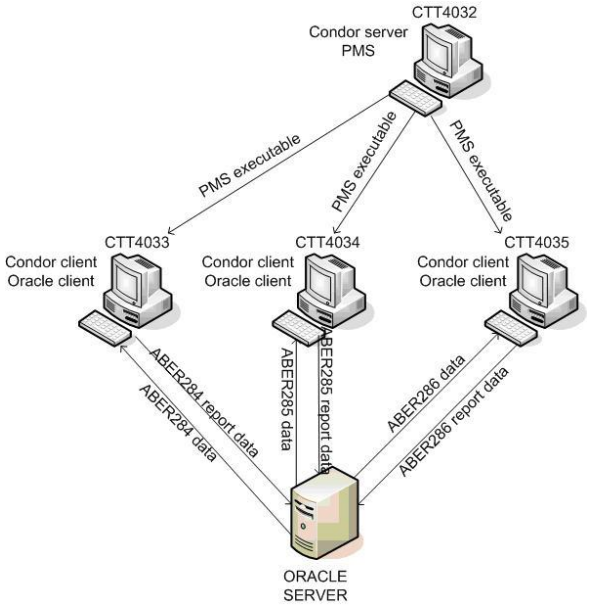
б) Време за получаване на план при алгоритъм със сортиране и генетичен алгоритъм

Фиг. 15. Времена за получаване на план за изпълнение от двата евристични алгоритъма при различен брой задачи

Резултатите показват експоненциалната сложност на алгоритъма с пълно изчерпване. Получаването на план с 9 задачи отнема на изчерпващия алгоритъм повече от 2 часа (фиг. 15а). Същият план се получава от алгоритъма със сортиране за около 1 минута, а от генетичния алгоритъм за 10 секунди (фиг. 15б). Следователно алгоритъмът с пълно изчерпване е практически неприложим при планиране на десет и повече задачи. Алгоритъмът със сортиране получава добър план за време, което е в порядъци по-малко от времето на изчерпващия алгоритъм. Най-добри са времевите характеристики на генетичния алгоритъм, но получените от него решения са по-лоши в сравнение с решенията на алгоритъма със сортиране, т.е. генерираният план има по-големи отклонения във времето за приключване на всички задачи в плана в сравнение с времето на съответния оптимален план.

3.4. Паралелно изпълнение на система за управление на портфейли

Проведени са експерименти с паралелно изпълнение на задачи за оценяване на фондове в разпределена среда, съставена от свързани компютри под управление на системата Condor и Oracle сървър (фиг. 16).



Фиг. 16. Среда за паралелно изпълнение на задачи за оценяване на фондове

Таблица 1. Обобщени данни за последователно и паралелно изпълнение

Задача / Изпълнение	Задача 1 (сек.)	Задача 2 (сек.)	Общо време (сек.)
Последователно	119	274	393
Паралелно без отложено стартиране (конкурентен достъп)	134	276	276
Паралелно с отложен старт на задача 2	161	269	269

Получените резултати представят ускорението, което се получава при отложено стартиране на задачите в разпределената среда.

4. Заключение

Предложеният в дисертационния труд подход използва естествения паралелизъм, характерен за много научни и приложни задачи. Чрез този подход могат да бъдат адаптирани за паралелно изпълнение програмни системи, решаващи такива задачи. Използването на подхода премахва необходимостта от сложни и скъпи преобразувания на програмния код. Задачите, за решаване на които може да бъде приложен подходът трябва да отговарят на следните условия:

- разделянето им на независими подзадачи не изисква специални техники или алгоритми;
- по време на изпълнението на задачите не се изисква взаимодействие с потребител.

Ако тези условия са изпълнени, задачите могат да бъдат изпълнени паралелно в локална мрежа или в GRID среда, съставена от налични обикновени работни станции.

Резултатите от проведените теоретични и експериментални изследвания определят следните приноси:

- Предложен е алгоритъм за паралелно изпълнение на независими задачи, при който предаваните съобщения между задачите са намалени до първоначално изпращане на данни към всеки изчислителен възел и събиране на крайните резултати от изчисленията.

- Разработени са два евристични алгоритъма за планиране на задачи в разпределена среда при ограничени ресурси.
- Разработени са програмни среди за тестване на предложените алгоритми. С помощта на тези среди е експериментално установено, че предложените евристични алгоритми получават близък до оптималния план за паралелно изпълнение на задачите за значително по-малко време в сравнение с изчерпващ алгоритъм.
- Формулиран е общ подход за изпълнение на паралелни задачи. Възможностите на подхода са демонстрирани при паралелно изпълнение на практически задачи за оценяване на риск в локална компютърна мрежа и в GRID среда.

Литература

[1]. Blazewicz, J. K. Lenstra. Scheduling Subject to Resource Constraints: Classification and Complexity. Discrete Applied Mathematics, 5, 1983, pp. 11-24.

[2]. Demeulemeester, E., W. Herroelen. A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. Management Science 38(12): 1803, 1992.

[3]. Dorndorf, U., E. Pesch, T. Phan-Huy. A Time-Oriented Branch-and-Bound Algorithm for Resource-Constrained Project Scheduling with Generalised Precedence Constraints. Management Science, Vol. 46, 10, 2000.

[4]. Pinedo, M. Scheduling: Theory, Algorithms, and Systems. ISBN: 978-0-387-78934-7, Springer Science+Business Media, LLC. New York, 2008.

[5]. Wall, M. B. A Genetic Algorithm for Resource-Constrained Scheduling. PhD thesis, submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements fot the degree of Doctor of Philosophy in Mechanical Engineering at the Massachusetts Institute of Technology, MIT, 1996.

[6]. Wilkinson, B., M. Allen. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice-Hall, ISBN: 0-13-671710-1, 1999.

[7]. Пенев, И. Моделиране на клас паралелни задачи чрез минимално оцветяване на граф. Научни трудове на Русенски университет, том 50, серия 3.2, ISSN: 1311-3321, 2011, стр. 212 – 217.

За контакти:
Ас. д-р инж. Ивайло П. Пенев
катедра „Компютърни науки и технологии”
Технически университет - Варна
E-mail: ivailo.penev@tu-varna.bg

СИНХРОНИЗАЦИЯ НА ТЕМПОТО ПРИ ЗАПИС НА MIDI-ФАЙЛ
(резюме на дисертация за получаване на образователна и научна степен "доктор")

Лъчезар Ил. Георгиев

Резюме: Предмет на даденото изследване са проблемите на загубата на темпото в процеса на запис на MIDI-файл. В резултат на изследването на основата на програмно-апаратна реализация на синтезатор на честота с автоматична донастройка на честотата и фазата е създаден метод за бързо и точно откриване на моментното темпо на възпроизвеждания MIDI-файл в процеса на записа му с външна MIDI-синхронизация. Той позволява да се възстанови първичната карта на темпата на възпроизвеждания файл с минимум грешни откривания на промени в темпото и може да се използва за преобразуване на произволна MIDI-секвенция в стандартен MIDI-файл.
Ключови думи: MIDI, файл, такт, темпо, синхронизация, запис.

Tempo Synchronisation Problems In MIDI File Recording
(summary of a dissertation for a doctor's degree)

Lutshayzar I. Gueorguieff

Abstract: The object of this research are the problems of tempo loss in MIDI file recording. The research resulted in creation of a method of fast and accurate estimation of the instant tempo of the MIDI file being played back during its recording with external MIDI clock. It permits restoration of the original tempo map of the played MIDI file with minimum false tempo change detections and is usable for conversion of any MIDI sequence into a standard MIDI file.
Keywords: MIDI, file, clock, tempo, synchronization, recording.

1. Увод

При запис на MIDI-данни, които се изпращат от възпроизвеждащо устройство, те трябва да се запишат така, че после да могат да се възпроизведат със същото темпо, с което са били възпроизведени по време на записа. Възпроизвеждащото устройство (секвенсер или компютър със секвенсерна програма, плейър, синтезатор, музикална работна станция, смесителен пулт, електронно пиано или орган) възпроизвежда файл, съдържащ MIDI-данни, но с нестандартен формат, разработен от фирмата-производител и прилаган само в нейните продукти. Записът става във формата на стандартен MIDI-файл (СМФ), който после може да се използва от всеки по-нов продукт – устройство като горепосочените или компютърна програма. Съществуващите устройства и програми не дават възможност за запазване на темпото на възпроизвеждане. Това е голям проблем при променливо темпо на музиката и пречка за пълното възстановяване на богатата музикална съкровищница от 80-те години на XX век, когато нямаше СМФ и MIDI-секвенциите в музикалните, филмови и видео-студия бяха създавани само във фирмения формат на използваното устройство или програма.

За решаване на този проблем бе създаден метод за автоматична синхронизация (съгласуване) на темпото на записвания СМФ към това на възпроизвеждания, по който бързо и точно се възстановява оригиналната последователност от темпа при синхронизирано възпроизвеждане. Той включва алгоритъм за откриване на моментното темпо при запис на изсвирван СМФ, който може да бъде приложен към произволна апаратна платформа, стига

тя да има два програмируеми таймера. Така може с достатъчна точност да се възстанови картата на темпата, което е особено полезно при променливо темпо.

2. Преобразуване на секвенсерен файл в СМФ

Съществуват три основни начина на преобразуване на секвенсерен файл от фирмен формат в СМФ – *пряко преобразуване* на файлове в СМФ, *прехвърляне на файлове* по MIDI (MIDI File Dump) и т. нар. „*синхронизирано просвирване*“ (synchronized over-play). За синхронизация на темпото може да се говори само при последния.

При **прякото преобразуване** на файлове се използва секвенсер или програма с такива възможности. При използване на секвенсер преобразуването става твърде бавно (доста под реалното време). Ако се използва програма за преобразуване, е необходима отделна програма за всеки фирмен формат. Фирмата „Giebler Enterprises“ предлага такива с обща стойност на програмите за 19 формата 19 × 55 = 1045 $. При това няма гаранции за качеството на преобразуването, тъй като информацията за файловите формати на секвенсерите на повечето фирми най-вероятно е получена чрез т. нар. „обратна разработка“ (reverse-engineering).

Прехвърлянето на файлове по MIDI е идеално решение на проблема за бездисково прехвърляне на СМФ. Но тъй като го поддържат много малък брой продукти, и то такива, които и без това поддържат СМФ, а секвенсерите, при които то би свършило работа, не го поддържат, то не може да реши проблема за преобразуването на фирмените секвенсерни файлове в СМФ.

За **синхронизирано възпроизвеждане** са необходими две устройства – възпроизвеждащо и записващо. Те се свързват с MIDI-кабел еднопосочно от изхода на първото към входа на второто. Устройствата се нагласят така, че възпроизвеждащото да предава MIDI-тактове (F8), а записващото да се синхронизира външно по тях чрез команди „старт“ и „стоп“. Накрая се пуска възпроизвеждащото устройство и записващото трябва да започне да записва, а на края на файла да спре по приетите команди „старт“ и „стоп“.

По този начин при нормално темпо на възпроизвеждане времетраенето на целия процес е равно на времетраенето на файла. Ако темпото на възпроизвеждане се повиши до максималното (обикновено до 200 – 250 удара на метронома в минута), процесът може значително да се ускори. Ако цялото произведение е в едно и също темпо, трябва след това в началото на така записания СМФ да се вмъкне едно мета-събитие за темпо, за да се получи СМФ, еднакъв с първоначалния. Следователно в този случай задачата за преобразуване на фирмен секвенсерен файл в СМФ се решава сравнително лесно.

Но при променливо темпо трябва да се идентифицират всички промени на темпото, тактът и времето на всяка промяна, и новото темпо, което се задава. След това трябва тези събития да се вмъкнат ръчно в записания СМФ. Понякога броят им може да достигне десетки или дори стотици, което забавя извънредно много тази процедура.

***Извод*:** До момента не съществува бърз и евтин метод за възстановяване на MIDI-темпата. Но описаната в по-горния абзац процедура може да се автоматизира. Това е задължително за работата ѝ при променливо темпо на музикалния материал.

3. Следене на MIDI-темпото с ФАДЧ

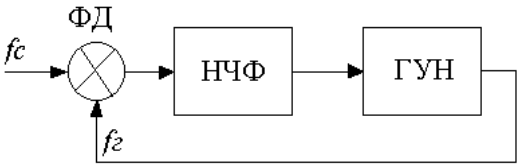
Задачата на настоящето изследване е създаването на метод за автоматично следене на темпото при запис на СМФ (вж. т. 1.3.1). MIDI-съобщенията за синхронизация (т. нар. „MIDI clock“) са с честота само 24 съобщения за четвъртинка нота, а времевата база (т. нар. „division“) е обикновено от 96 до 480 кванта от време за четвъртинка нота (т. е. 4 до 20-кратно по-висока) и *винаги* кратна на 24. За да се запишат делта-времената на MIDI-събитията, е

необходим вътрешен тактов генератор в записващото устройство, който да работи на честотата на времевата база, т. е. с многократно по-висока честота от тази на MIDI-съобщенията за синхронизация. Как може да стане това? Следващите подточки съдържат отговор на този въпрос.

3.1. Използване честотен синтезатор за времева база

Предлага се да се използва програмно-управляем честотен синтезатор, чийто делител на честота е един от програмируемите таймери на компютъра. Коефициентът му на делене се управлява по програмен път чрез система за фазова автодонастройка на честотата (ФАДЧ) или „Phase-locked-loop“ (PLL) в англоезичната литература. По-долу е дадена съвсем накратко теорията на такъв един честотен синтезатор.

На фиг. 1 е показана блоковата схема на система за ФАДЧ. Фазовият детектор ФД има за цел да открие наличието на разлика във фазите на двата сигнала. Това е възможно само ако честотите на двата сигнала са приблизително еднакви. При наличието на фазова разлика, на изхода на фазовия детектор трябва да се получи сигнал, чиито параметри да се изменят пропорционално на тази фазова разлика.



Фиг. 1. Структурна схема на система за ФАДЧ

На фиг. 2 е дадена блокова схема на синтезатор с един делител на честотата. Делителят се включва между управляемия генератор и фазовия детектор. Той дели честотата на управляемия генератор с коефициент, зададен на делителя по програма. В резултат на действието на системата за ФАДЧ във всеки момент трябва

$$f_{вх} = f_д \tag{1}$$

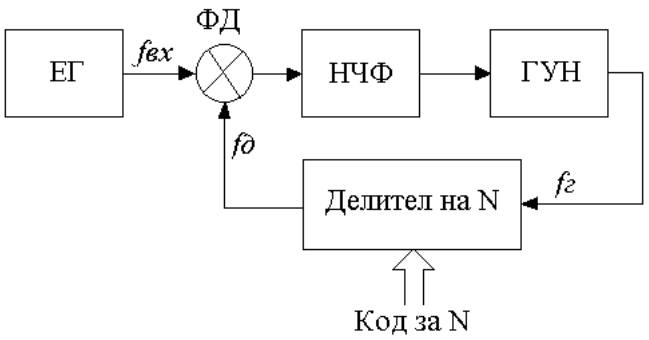
където $f_{вх}$ е честотата на кварцовия генератор, а $f_д$ – на сигналите след делителя. Но

$$f_д = \frac{f_г}{N} \tag{2}$$

където $f_г$ е изходната честота на управлявания генератор. Следователно,

$$f_г = Nf_{вх} \tag{3}$$

Чрез промяната на N изходната честота на генератора се променя със стъпка $f_{вх}$.



Фиг. 2. Схема на синтезатор на честота с един делител

В конкретния случай еталонен генератор са MIDI-съобщенията за синхронизация, управляем генератор – времевата база, а делител – програмируемият таймер.

Заб.: Съществуват и синтезатори с дробен коефициент на делене, но тук не се използва такъв, защото честотата на управляемия генератор (в този случай от 48 до 480 / ♩ през 24) е винаги кратна на тази на еталонния генератор (в този случай 24 / ♩).

3.2. Измерване на честотата на синхробайтовете

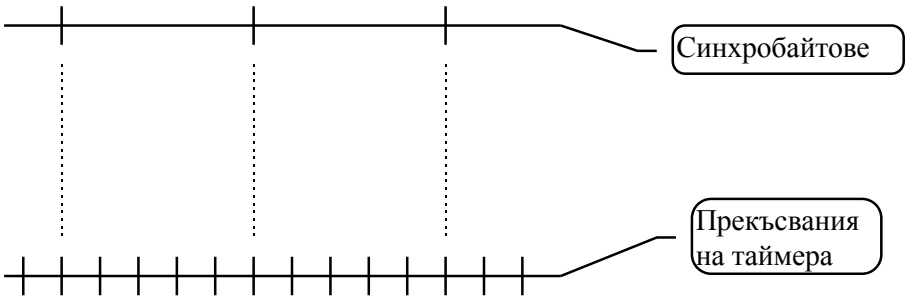
Коефициентът на делене на програмируемия делител на честота трябва да е пропорционален на честотата на синхробайтовете на входа, която пък може да се измери с втори таймер. За разлика от таймера на честотния синтезатор, който работи в режим на програмируем делител на честота, този таймер работи в режим на измерване на честота.

4. Алгоритми за възстановяване на MIDI-темпото

В следващите две подточки първо е описан принципът на действие, а след това са описани и алгоритмите на предложения метод за автоматично възстановяване на картата на темпата при запис на MIDI-файлове с външна MIDI-синхронизация.

4.1. Принцип на действие

Този метод може да бъде приложен към произволна апаратна платформа, стига тя да има два програмируеми таймера. Таймер №1 измерва честотата на входящия поток синхробайтове с точност ≤ 1 μs. С една пета от измерената стойност се инициализира таймер №2, за да генерира прекъсвания с честота, 5 пъти по-висока от честотата на синхробайтовете, т. е. прави се автоматичната донастройка на честотата от т. 3.1. Програмата за обслужване на прекъсването увеличава с 1 брояч, който непрекъснато се сверява с брояча на синхробайтовете, и ако избързва или изостава, се намалява или увеличава (програмна реализация на фазов детектор). Ако два или повече синхробайта дойдат наведнъж, което става след системни специални съобщения, по-дълги от 1/96 нота (24 × 4 = 96), настройката се забранява до нормализиране на синхропотока („шумоподтискане“).



Фиг. 3. Съответствие между синхробайтове и прекъсвания на таймера

Измерената честота на синхробайтовете се усреднява до знаменателя на метричния размер (най-често осминка или четвъртинка нота, т. е. върху 12 или 24 интервала на дискретизация). Тази средна стойност се сравнява с текущата, която междувременно постоянно се настройва (вж. по-горе). Ако разликата е над прага на различимост на промяна в темпото (2%), за времето на последната промяна се вмъква мета-събитие „Темпо“ с последната отчетена стойност. Тестовете показват, че ако една пиеса има постоянно темпо, така се създава само едно събитие „Темпо“.

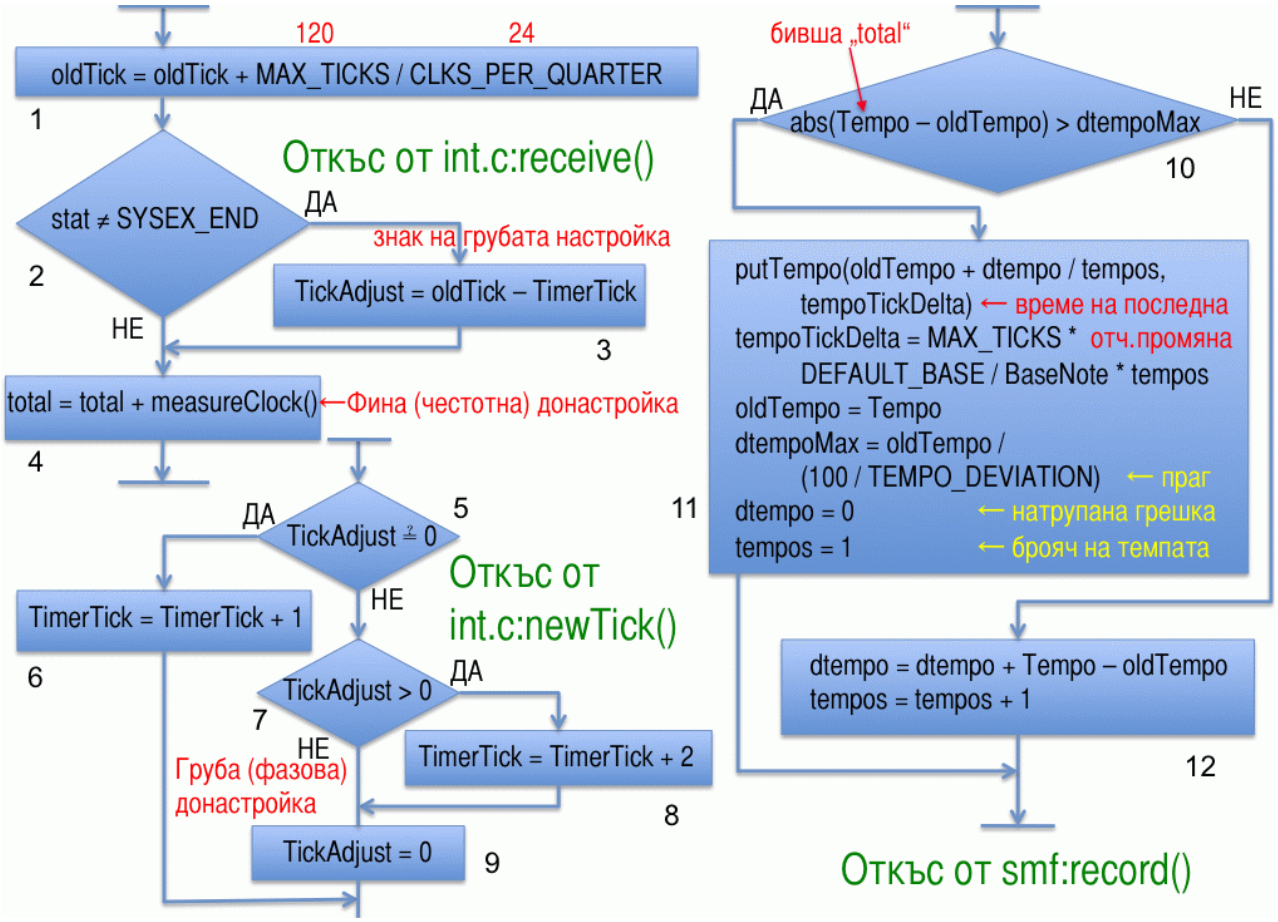
4.2. Описание на алгоритмите

Блокова схема на алгоритмите е дадена на фиг. 4, а на фиг. 3 е показано разположението на синхробайтовете и прекъсванията на таймера във времето. Вижда се, че последните са с пет пъти по-висока честота. Това е така, защото в една четвъртинка нота по стандарт има 24 синхробайта, а прекъсванията на таймера са 120 за четвъртинка нота, с цел по-висока разделителна способност.

В блок 1–4 от фиг. 4 е показан откъс от функцията за обслужване на прекъсването по приети данни от второ ниво *int.c:receive()*. Този участък се изпълнява само ако приетият байт е синхробайт. Броячът *oldTick* регистрира таймерните прекъсвания, които би трябвало да са вече обработени при постъпване на синхробайта. *MAX_TICKS* е броят на прекъсванията за една четвъртинка нота и е установен на 120. *CLKS_PER_QUARTER* е броят на синхробайтовете за четвъртинка, по стандарт 24. Така между всеки два синхробайта има 5 таймерни прекъсвания; с толкова се увеличава и *oldTick*.

В блок 3 глобалната променлива *TickAdjust* регистрира знака на грубата (фазова) настройка. Тя се прави при всяко таймерно прекъсване, ако *TickAdjust* е ненулева. В края на системните специални съобщения настройката се забранява (блок 2). Фината (честотна) настройка става в блок 4. Променливата *total* регистрира интервала от време между две таймерни прекъсвания, получен чрез функцията *measureClock()*. Тази функция е написана на асемблер и използва първия таймер, за да измери периода между два синхробайта, след което инициализира втория таймер със стойност, една пета от измерената. Така между два синхробайта винаги има 5 прекъсвания на втория (системен) таймер.

По-късно *total* се използва за инициализация на глобалната променлива *Tempo*.



Фиг. 4. Алгоритми за възстановяване на темпата. Сложността им е O(1) поради липса на цикли.

В блокове 5–9 е показана основната част от функцията *int.c:newTick()* за обновяване на брояча на таймерните прекъсвания *TimerTick*. Тази функция се извиква при всяко прекъсване на системния таймер. В нея се извършва и грубата (фазова) настройка на брояча. Това става само с една стъпка напред или назад, за да се избегнат резките промени, които биха довели до генериране на излишни мета-събития „Темпо“.

В блокове 10–12 е показан участък от кода на функцията за запис *smf:record()*. Ако разликата в темпото е по-голяма от прага *dTempoMax*, се генерира събитие за темпо със стойност, отчитаща натрупаната грешка *dtempo*, по времето на последната отчетена промяна на темпото *tempoTickDelta*, след което това време, темпото и прагът се актуализират, а грешката и броячът на темпата *tempos* се инициализират. В противен случай просто се натрупва грешката и се увеличава броячът. Накрая на записа се генерира и последното събитие за темпо (това не е показано на схемата). Така при пиеса с постоянно темпо се създава само едно събитие за темпо.

5. Тестове на разработения метод за възстановяване на темпата

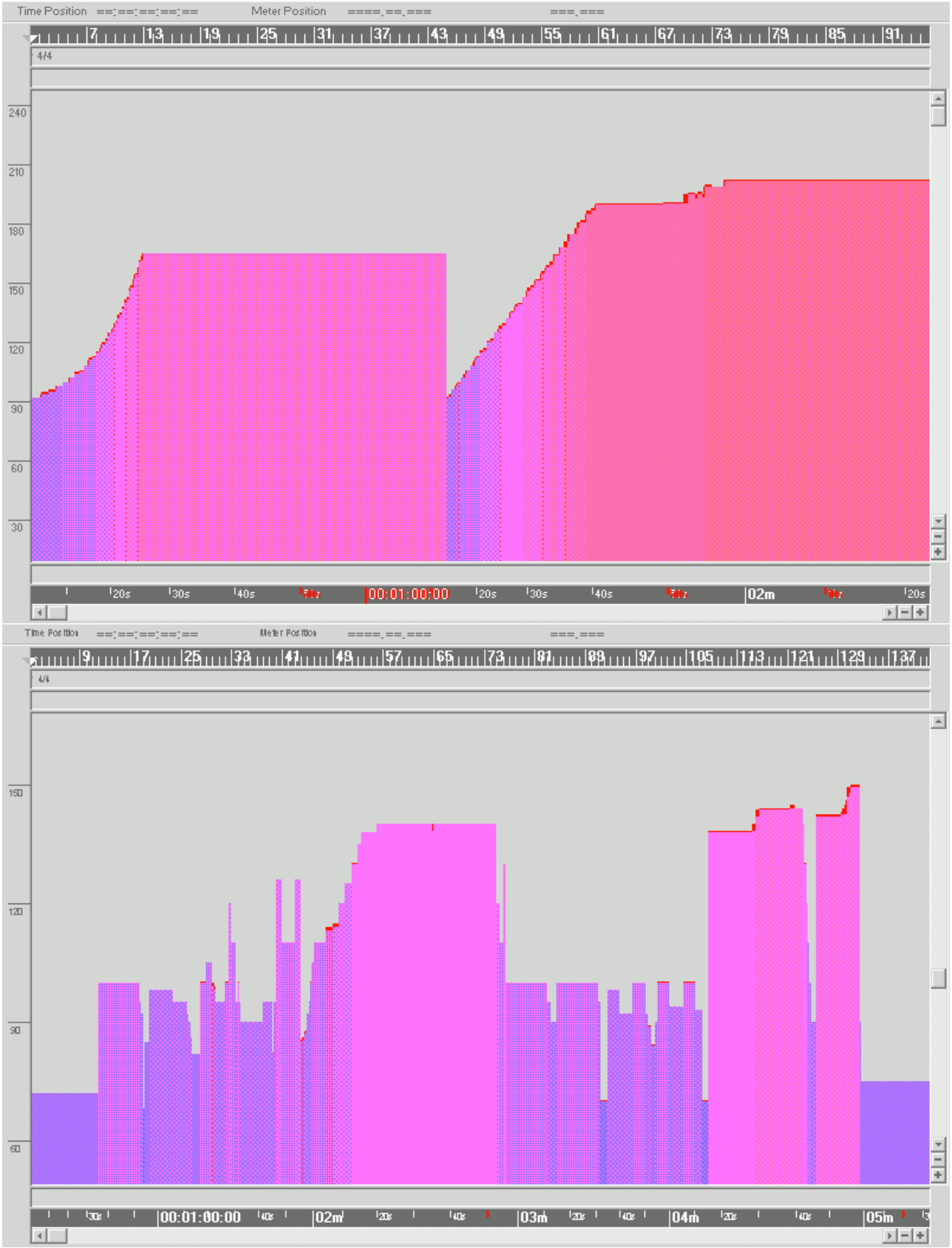
При изпитанията на метода с помощта на устройството „MFP-02“, в което е внедрен, се потвърди неговата ефективност, бързина и точност. Използвани бяха пиеси със сложна и динамична карта на темпата или съдържащи системни специални съобщения, затрудняващи възстановяването. Ето средностатистическите грешки, получени в резултат на измерванията:

пиеса	мин:сек	dtempo	dtime	m	n	$\overline{x}$	σ_n	σ_{n-1}
arabesq	0:36,90	—	0,035%	76	70	0,019%	0,132%	0,133%
baklava	2:00,24	—	0,052%	145	53	—	—	—
clairdln	1:04,19	—	1,240%	41	41	0,005%	0,011%	0,011%
czardas	5:04,41	—	0,018%	88	81	0,009%	0,234%	0,236%
furelise	0:57,73	—	0,013%	447	128	—	—	—
italco	3:46,43	—	0,018%	387	293	—	—	—
minuet1	0:35,88	0,034%	0,017%	1	1	—	—	—
sc55all	0:05,77	—	4,832%	1	11	–0,092%	3,883%	4,073%

Пиесите в първата колона от горната таблица могат да се характеризират, както следва:

- **arabesq** – пиеса за пиано, темпо рубато, променящо се на всяка четвъртинка нота.
- **baklava** – руска народна песен, напълно оркестрирана, динамично темпо.
- **clairdln** – класическа пиеса за пиано, много бавно темпо, възпроизведена в ДОС.
- **czardas** – откъс от оперетата „Царицата на чардаша“, много динамично темпо.
- **furelise** – „За Елиза“, класическа пиеса за пиано от Лудвиг ван Бетховен.
- **italco** – класическа пиеса за пиано, темпото се променя на всяка четвъртинка нота.
- **minuet1**– класическа пиеса за пиано (менует, такт 3/4), постоянно темпо.
- **sc55all** – съдържа само системни специални съобщения средно от по 256 байта.

Графата *dtempo* показва грешката в измереното темпо и е запълнена само за „*minuet1*“, защото само тя има постоянно темпо; *dtime* съдържа грешката във времетраенето на пиесата; *m* е броят на оригиналните, а *n* – на записаните събития „Темпо“, а $\overline{x}$, σ_n и σ_{n-1} са средно-аритметичната грешка за всяко събитие „Темпо“ и средните ѝ стандартни отклонения (последното за доверителен интервал 100%). Последните три графи са попълнени само при съвпадение на броя на събитията „Темпо“ в оригинала и в записаното копие или при съседни събития „Темпо“ с близки стойности.



Фиг. 5. Разлики (оцветени в яркочервено) между оригиналната и възстановената при запис карта на темпата за песните „baklava“ (горе) и „czardas“ (долу)

Ниската стойност на грешката при „*clairdln*“ се получава, защото е възпроизведена в ДОС, а не във „Windows“ като другите, а голямата грешка във времетраенето – защото възпроизвеждащата я програма забавя началото на песента, за да нулира MIDI-контролерите. В нея и в „*arabesq*“ има резки промени на темпото и именно там то се възстановява най-точно.

Случаят със „*sc55all*“ е екстремен, защото „нормалните“ песни съдържат системни специални съобщения само в началото, а по средата на песента, ако ги има, те са много къси. Затова такава голяма грешка (3–4%) не може да се получи в практиката. За отбелязване е, че въпреки постоянното темпо на „*sc55all*“ се генерират 10 фалшиви събития „Темпо“ поради ниския праг (2%). Ако прагът се вдигне до 5%, това няма да става, но при „нормална“ музика ще се пропускат по-малките промени в темпото.

На фиг. 5 са показани разликите в оригиналната и възстановената при запис карта на темпата на пиесите „*baklava*“ и „*czardas*“, избрани заради някои аномалии при възстановяването, които не се наблюдават при останалите и са описани по-подробно тук.

„*Baklava*“: Между тактове 49 и 61 не всички промени на темпото са регистрирани, а между тактове 67 и 73 има излишни и неточни събития „Темпо“. Тези аномалии произтичат от стойността на прага (в случая 2%). В оригиналната пиеса има малки промени на темпото, които се пропускат при възстановяването, защото остават под прага, но причиняват грешки при усредняването на следващите 1–2 такта, особено при такт 72 (фиг. 5).

„*Czardas*“**:** При тактове 65 и 121 има две фалшиви събития „Темпо“, съответно с по-висока и с по-ниска стойност от нормалната. Те са се получили поради неравномерността на синхробайтовете при възпроизвеждане под ОС „Windows 9x“ – там мултимедийната задача не е с най-високия възможен приоритет, както е при семейството „NT“. При по-бърз компютър или по-добра ОС тези флуктуации ще бъдат под защитния праг.

6. Изводи и практическа реализация

Както личи от гореспоменатите измервания, методът дава добри резултати дори и при сложна карта на темпата с резки промени (най-важни за слуховъзприятието).

Възстановяването на картата на темпата позволява да се съхрани цялата времева информация с музикална стойност по време на записа – както реалните позиции и продължителността на нотите и разделението по тактове, така и промените в темпото по време на музикалното изпълнение, съчетавайки предимствата на синхронизацията по синхробайтове (MIDI-„clock“ = MIDI-„часовник“) и по код за време (MTC, „MIDI Time Code“ = MIDI-код за време), т. е. съответно по относително и абсолютно време.

Фиг. 6. Устройство за запис, редактиране и възпроизвеждане на MIDI-файлове „MFP-03“ в действие (текущо темпо при възпроизвеждане: ♩ = 193)

Методът за възстановяване на картата на темпата бе внедрен в устройствата за запис, редактиране и възпроизвеждане на MIDI-файлове „MFP-02“ и „MFP-03“ (фиг. 6).

7. Заключение

Проблемите на синхронизацията, които възникват при запис на MIDI-файлове могат успешно да бъдат решени. Проблемът се свежда до откриване и регистриране на текущото темпо с достатъчна точност, бързина и устойчивост (липса на колебания, т. е. на излишни събития „Темпо“). Този въпрос до момента не е решен. Възможно решение за бързо и точно *следене на моментното темпо* на просвирвания СМФ при записа му с външна MIDI-синхронизация е предложено в настоящото изследване. То е на базата на честотен синтезатор с автоматична донастройка на честотата и фазата, осъществена по програмен път, която използва два програмируеми таймера. Изпитанията показват, че с нея успешно се възстановява първичната карта на темпата на възпроизвеждания файл с минимум грешни откривания на промени в темпото. Това дава възможност да се преобразуват в MIDI-файлове всички секвенции, които са записани в един или друг фирмен формат на по-стари устройства, които не поддържат или не използват СМФ като свой вътрешен формат, чрез запис с външна MIDI-синхронизация върху устройство, прилагащо предложения метод. За разлика от известните методи този метод е универсален, т. е. с него може да се преобразува произволна MIDI-секвенция в СМФ.

Възможна насока за бъдещо развитие на изследванията е автоматичното регулиране на прага за откриване на събитие „Темпо“.

Литература

[1]. Георгиев, Л., „Метод за автоматично възстановяване на картата на темпото при запис със синхронизация по MIDI-часовник“, Национална конференция с международно участие „Телеком ’97“, т. I, Доклади на български език, 1998 г., стр. 574–580.

[2]. Gueorguieff, L., Antonov, P., “Tempo Map Retrieval from the MIDI Clock Stream”, Proceedings of the XLVIII International Scientific Conference on Information, Communication and Energy Systems and Technologies ICEST 2013, vol. 1, pp. 145–148.

За контакти:
гл. ас. д-р Лъчезар И. Георгиев
Катедра „Компютърни науки и технологии”
Технически университет – Варна
email: lucho@mbox.contact.bg

ИЗИСКВАНИЯ ЗА ОФОРМЯНЕ НА СТАТИИТЕ ЗА СПИСАНИЕ "КОМПЮТЪРНИ НАУКИ И ТЕХНОЛОГИИ”

I. Статиите се представят разпечатани в два екземпляра (оригинал и копие) в размер до 6 страници, формат А4 на адрес: Технически университет – Варна, ФИТА, ул. „Студентска” 1, 9010 Варна, както и в електронен вид на имейл адреси: peter.antonov@ieee.bg или jppet@abv.bg.

II. Текстът на статията трябва да включва: УВОД (поставяне на задачата), ИЗЛОЖЕНИЕ (изпълнение на задачата), ЗАКЛЮЧЕНИЕ (получени резултати), БЛАГОДАРНОСТИ към сътрудниците, които не са съавтори на ръкописа (ако има такива), ЛИТЕРАТУРА и информация за контакти, включваща: научно звание и степен, име, организация, поделение (катедра), e-mail адрес.

III. Всички математически формули трябва да са написани ясно и четливо (препоръчва се използуване на Microsoft Equation).

IV. Текстът трябва да бъде въведен във файл във формат WinWord 2000/2003 с шрифт Times New Roman. Форматирането трябва да бъде както следва:

1. Размер на листа - А4, полета: ляво - 20мм, дясно - 20мм, горно - 15мм, долно - 35мм, Header 12.5мм, Footer 12.5мм (1.25см).
2. Заглавие на български език - размер на шрифта 16, удебелен, главни букви.
3. Един празен ред - размер на шрифта 14, нормален.
4. Имена на авторите - име, инициали на презиме, фамилия, без звания и научни степени - размер на шрифта 14, нормален.
5. Два празни реда - размер на шрифта 14, нормален.
6. Резюме и ключови думи на български език, до 8 реда - размер на шрифта 11, нормален.
7. Два празни реда - размер на шрифта 11, нормален.
8. Заглавие на английски език - размер на шрифта 12, удебелен.
9. Един празен ред - размер на шрифта 11, нормален.
10. Имена на авторите на английски език - размер на шрифта 11, нормален.
11. Един празен ред - размер на шрифта 11, нормален.
12. Резюме и ключови думи на английски език, до 8 реда - размер на шрифта 11, нормален.
13. Основните раздели на статията (Увод, Изложение, Заключение, Благодарности, Литература) се форматират в едноколонен текст както следва:
    a. Наименование на раздел или на подраздел - размер на шрифта 12, удебелен, центриран, един празен ред преди наименованието и един празен ред след него - размер на шрифта 12, нормален;
    b. Текст - размер на шрифта 12, нормален, отстъп на първи ред на параграф – 10 мм; разстояние от параграф до съседните (Before и After) за целия текст – 0.
    c. Цитиране на литературен източник - номер на източника от списъка в квадратни скоби;
    d. Текстът на формулите се позиционира в средата на реда. Номерация на формулите - дясно подравнена, в кръгли скоби.
    e. Фигури - центрирани, разположение спрямо текста: “Layout: In line with text”. Номер и наименование на фигурата - размер на шрифта 11, нормален, центриран. Отстояние от съседните параграфи – 6 pt.
    f. Литература – всеки литературен източник се представя с: номер в квадратни скоби и точка, списък на авторите (първият автор започва с фамилия, останалите – с име), заглавие, издателство, град, година на издаване, страници.
    g. За контакти: научно звание и степен, име, презиме (инициали), фамилия, организация, поделение (катедра), e-mail адрес, с шрифт 11, дясно подравнено.

Образец за форматиране можете да изтеглите от адрес http://cs.tu-varna.bg/ - Списание КНТ, Spisanie_Obrazec.zip.

Този брой от списанието се отпечатва със средства от научно-изследователски проект, финансиран целево от държавния бюджет.