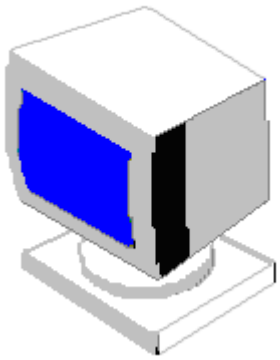
ТЕХНИЧЕСКИ
УНИВЕРСИТЕТ
ВАРНА

КОМПЮТЪРНИ НАУКИ
и
ТЕХНОЛОГИИ

година IV брой 1/2006

COMPUTER SCIENCE
AND
TECHNOLOGIES



IEEE
Bulgaria
Communications
Chapter

TU-Varna
FCA
Faculty of Computing & Automation

Компютърни науки и технологии

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

Редактор: доц. д-р О. Железов
Гл. редактор: доц. д-р П. Антонов

Печат: ТУ-Варна

За контакти:

Технически университет - Варна
ул. Студентска 1, 9010 Варна, България
тел/факс: (052) 383 320
e-mail: peter.antonov@ieee.org
ognyanz@yahoo.com

ISSN 1312-3335

Computer Science AND Technologies

Publication
of Computing and Automation Faculty
Technical University of Varna

Editor: Assoc. Prof. O. Zhelezov
Chief Editor: Assoc. Prof. P. Antonov

Printing: TU-Varna

For contacts:

Technical University of Varna
1, Studentska Str., 9010 Varna, Bulgaria
Tel/Fax: (+359) 52 383 320
e-mail: peter.antonov@ieee.org
ognyanz@yahoo.com

ISSN 1312-3335

СЪДЪРЖАНИЕ

CONTENTS

Информационни технологии I Information Technologies

СЪДЪРЖАНИЕ

CONTENTS

НАДЕЖДНА КОМУНИКАЦИЯ МЕЖДУ МИКРОКОНТРОЛЕР И ИНТЕЛИГЕНТНИ ДАТЧИЦИ

Драгомир Х. Славов, Милена Я. Караиванова, Петър Ц. Антонов

Резюме: Представя се разработена и внедрена система за измерване на температура, налягане и влажност на базата на микроконтролер dsPIC33FJ256GP710 и интелигентни датчици MS5540B и SHT15. За осигуряване на надеждна комуникация между микроконтролера и датчиците в системата е разработен ефективен програмен алгоритъм за защита от грешки с използване на цикличен код и обратна връзка.

A Reliable Communication Between Microcontroller and Intelligent Sensors

Dragomir H. Slavov, Milena J. Karaivanova, Peter T. Antonov

Abstract: This paper reviews an incorporated system for temperature, pressure and humidity measuring on the basis of the microcontroller dsPIC33FJ256GP710 and intelligent sensors MS5540B and SHT15. A software error protection algorithm for insurance of the reliable communication between the microcontroller and sensors in the system, by using a cyclic code and feedback, is presented too.

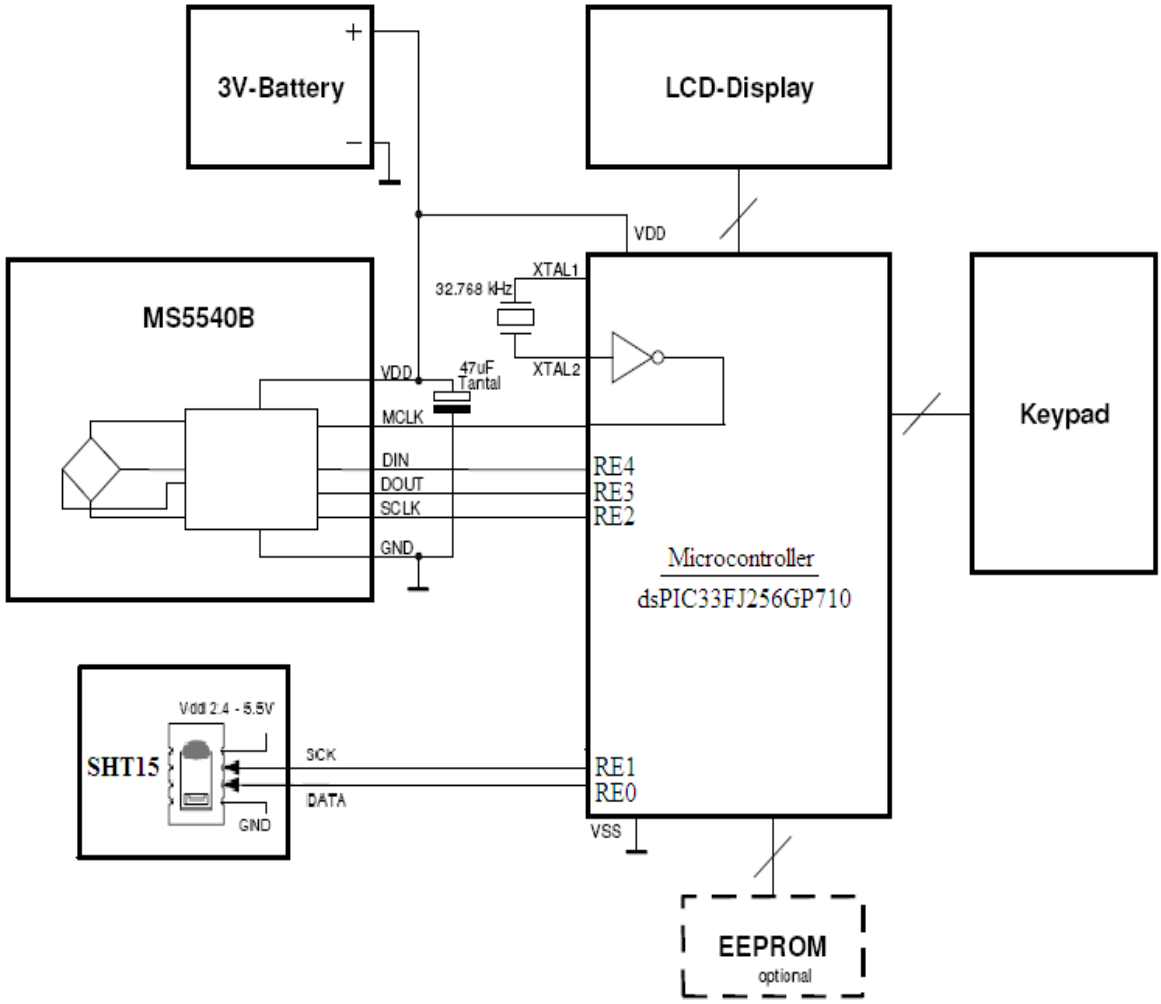
1. Описание на системата

Апаратната част на разработената система за измерване на температура, налягане и влажност, базирана на интелигентни датчици и микроконтролер dsPIC 33FJ256GP710, е представена на фиг. 1.

За свързване на датчика за измерване на температура и налягане MS5540B към контролера се използват три линии (DIN, DOUT и SCLK) и крачета 100(RE4), 99(RE3) и 98(RE2). На линия DIN се поставя последователност, която съответства на подадена за изпълнение команда (четене на данни за температура, налягане и т.н.). След получаване на командата, датчикът я изпълнява и поставя на линията DOUT исканите данни, които се прочитат в съответствие със сигнала на линия SCLK.

За свързване на датчика за измерване на температура и влажност SHT15 се използват крачета 94(RE1) и 93(RE0). Комуникацията между този датчик и микроконтролера се осъществява посредством две линии – SCK и DATA. На линия DATA се поставя зададената от микроконтролера команда (четене на данни за температура, влажност и т.н.), както и резултата от изпълнената от датчика команда. Командата и върнатият резултат се прочитат при сигнал на линия SCK.

Програмното осигуряване на системата е разработено с използване на развойна среда MPLab C30 с Cross C компилатор и развоен кит Explorer 16. То включва програми за реализация на необходимите за целта функции за комуникация между двата датчика и микроконтролера – синхронизация, изпращане на команда към всеки един от датчиците, задръжка за обработка на данни, четене на данни от отделните датчици и др.



Фиг. 1.

2. Постановка на задачата

Включеният в системата датчик за измерване на температура и влажност SHT15 поддържа възможност за повишаване на надеждността на комуникацията с използване на цикличен код (CRC - Cyclic Redundance Code). Както е известно, този тип шумоустойчив код се характеризира с найдобра способност за откриване на грешки, което определя и широката му приложимост в практиката [1,2 и др.].

След като датчикът изпрати данните, поискани му от микроконтролера, той изпраща и CRC контролна сума. Тази сума се изчислява на базата на получената от датчика команда и изпращаните от него данни, по известната формула [1,2,3 и др.]

$$\frac{P(x).x^r}{G(x)} = Q(x) + \frac{R(x)}{G(x)}, \qquad (1)$$

където

$P(x)$ - полиномен запис на получената команда и на изпращаните от датчика данни;

$G(x)$ - генераторен полином на цикличния код (r – степен на генераторния полином и брой на контролните разряди);

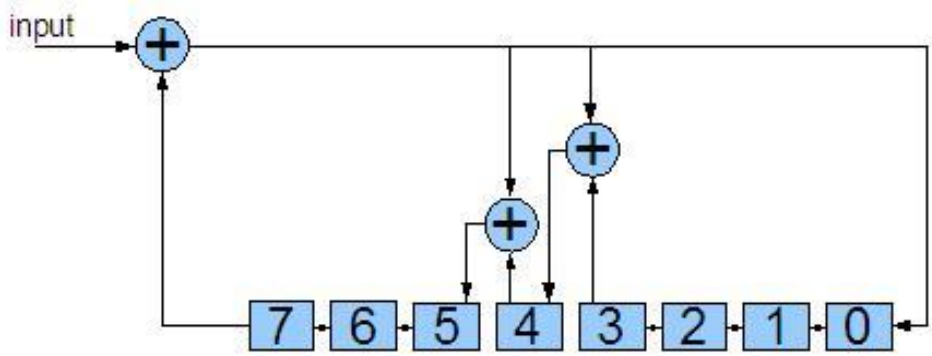
$R(x)$ - CRC контролна сума.

Командата и данните, с обща дължина от 24 бита, се представят като полином $P(x)$ от 23-та степен. Броят на контролните разряди, определен от степента на генераторния полином, е 8.

Форматът на командата с данните и контролната сума е представен на фиг. 2, а

схемата за апаратно изчисляване на CRC контролната сума от датчика - на фиг. 3.

Фиг. 2.



Фиг. 3.

По същество, схемата на фиг. 3 представлява преместващ регистър с обратни връзки, който реализира деление на постъпващия на входа полином с използвания в случая генераторен полином $G(x) = x^8 + x^5 + x^4 + 1$. Съдържанието на регистъра, след делението, съответства на остатъка $R(x)$, който се приема за контролна сума и се изпраща от датчика към микроконтролера, който от своя страна следва да направи проверка за наличието на грешки.

В тази връзка, целта на по-нататъшното изложение се изразява в разработката на ефективен алгоритъм за контрол на грешките от страна на микроконтролера.

3. Обосновка на алгоритъма

От фиг. 2 следва, че използваният в случая цикличен код може да бъде записан като CRC(32,24). Кодовите комбинации са с дължина 32 бита и могат да се представят чрез полиноми от 31-ва степен. Спецификата тук се изразява в това, че командата не се връща обратно от датчика, тъй като се помни от микроконтролера, но се използва задължително за изчисляване на контролната сума.

На базата на изпратената команда и на получените данни от датчика, микроконтролерът може да изчисли контролна сума и да я сравни с изпратената от датчика. Различието между изчислената и получената контролни суми означава наличието на грешки и тогава микроконтролерът следва да изпрати отново същата команда към датчика за повторно изпълнение.

За изчисляване на контролната сума в микроконтролера може да се използва софтуерна симулация на работата на схемата от фиг. 3. В случая, обаче, може да се предложи по-ефективен подход, базиран на спецификата на цикличния код и на конкретния формат от фиг. 2.

Ако с $P_1(x)$ отбележим полиномния запис на кода на командата, с $P_2(x)$ - полиномния запис на първия байт на данните и с $P_3(x)$ - полиномния запис на втория байт на данните, то

$$P(x) = P_1(x).x^{16} + P_2(x).x^8 + P_3(x). \qquad (2)$$

При това полагане формула (1) може да се запише в случая по следния начин:

$$\frac{P_1(x).x^8}{G(x)}.x^{16} + \frac{P_2(x).x^8}{G(x)}.x^8 + \frac{P_3(x).x^8}{G(x)} =$$

$$= Q_1(x).x^{16} + \frac{R_1(x).x^{16}}{G(x)} + \frac{P_2(x).x^{16}}{G(x)} + \frac{P_3(x).x^8}{G(x)} =$$

$$= Q_1(x).x^{16} + \frac{[R_1(x) + P_2(x)].x^8}{G(x)}.x^8 + \frac{P_3(x).x^8}{G(x)} =$$

$$= Q_1(x).x^{16} + Q_2(x).x^8 + \frac{R_2(x)}{G(x)}.x^8 + \frac{P_3(x).x^8}{G(x)} =$$

$$= Q_1(x).x^{16} + Q_2(x).x^8 + \frac{[R_2(x) + P_3(x)].x^8}{G(x)} =$$

$$= Q_1(x).x^{16} + Q_2(x).x^8 + Q_3(x) + \frac{R_3(x)}{G(x)}.$$

От горния запис може да се изведе следния алгоритъм: полиномният запис $P_1(x)$ на кода на командата се умножава по x^8 и се дели на $G(x)$; към получения остатък $R_1(x)$ се прибавя полиномния запис на първия байт на данните $P_2(x)$; полученият резултат
се умножава по x^8 и се дели на $G(x)$; остатъкът от делението $R_2(x)$ се сумира с полиномния запис на втория байт на данните $P_3(x)$, резултатът също се умно-жава с x^8 и се дели на $G(x)$; полученият от последното деление остатък $R_3(x)$ се явява и търсената контролна сума $R(x)$. При практическата реализация на този алгоритъм първото деление може да се направи предварително за всички възможни команди. Тогава, съответните остатъци $R_1(x)$ ще се вземат директно от таблица и по същество ще се реализира само второто и третото деления.

4. Реализация на алгоритъма

Представеният по-горе алгоритъм за изчисляване на контролните суми при формата от фиг. 2 е много удобен за реализация от 8-битови микропроцесори. В подкрепа на това, по-долу е приведена програма за MC6800, където: ИБ – информационен байт, а КБ - контролен байт (контролна сума). Програмата обхваща по-общия случай на N информационни байта, като N може да се променя до $30 = 1E_{16}$. В случая, конкретната стойност на N се въвежда в клетка 0021, в клетките от 0001 до 001E се записват информационните байтове, в клетка 001F - изчислената 8-битова контролна сума, а в клетка 0000 - генераторния полином, но без старшия бит, който винаги е 1 и се подразбира.

```
0000        > G(x) + x^8
0001        > 1ИБ
0002        > 2ИБ
.
.
001E        > NИБ
001F        > 0    (Клетка за КБ)

0020 00     Старт
0021        > N
0022 CE
0023 00
0024 01
0025 A8
0026 00
0027 8D     Преход към
0028 0B     подпрограма
0029 9C
002A 20
002B 27
002C 03
002D 08
002E 20
002F F5
0030 08
0031 A7     Запис на
0032 00     КБ
0033 3F     Стоп

0034 C6
0035 08
0036 48
0037 24
0038 02
0039 98
003A 00
```

003B 5A
003C 2E
003D F8
003E 39

Делениятa на генераторния полином се реализират с използване на флага C и двата акумулатора A и B. Първоначално в A се записва първия ИБ. Реализира се ляво преместване и анализ на флага C, в който се намира старшия бит на 1ИБ. При (C)=1 към съдържанието на A се прибавя по *mod2* $G(x)+x^8$, след което се прави следващо ляво преместване. При (C)=0 – се осъществява само следващо преместване. След 8 такива премествания в A се получава първия остатък от делението. Към него се прибавя 2ИБ, прави се ляво преместване и т.н.

В конкретния случай *N*=3 и първите остатъци $R_1(x)$ се определят предварително за всяка от командите. Тогава, към $R_1(x)$ следва да се прибави получения от датчика $P_2(x)$, след което да се реализират последователно 8 леви премествания с анализ на C и сумиране с $G(x)+x^8$, при (C)=1. Полученият остатък $R_2(x)$ трябва да се сумира с $P_3(x)$ и да се реализират нови 8 премествания с анализ на C и евентуално сумиране с $G(x)+x^8$. Като резултат ще се получи изчислената контролна сума $R(x)$, която следва да се сравни с изпратената такава от датчика. Този алгоритъм е реализиран на C и за използвания в разработената система микроконтролер.

В заключение ще отбележим, че е възможно допълнително усъвършенстване на алгоритъма с цел повишаване на бързодействието. За целта, подобно на $R_1(x)$, остатъците $R_2(x)$ и $R_3(x)$ могат също да бъдат предварително изчислени за всеки възможен байт и записани в таблица. При това положение изчисляването на контролнара сума ще става много бързо, тъй като по $[R_1(x)+P_2(x)]$ от таблицата веднага ще се извлече $R_2(x)$, а по $[R_2(x)+P_3(x)]$ - търсената контролна сума $R(x)$.

Литература

[1].Касами, Т. и др. Теория кодирования. Пер. с япон. – М., Мир, 1978. – 576 с.

[2].Питерсон, У., Э. Уелдон. Коды, исправляющие ошибки. Пер. с англ. – М., Мир, 1976. – 594 с.

[3]. Halsall, F. Multimedia Communications. Applications, Networks, Protocols and Standards. – Personal Education Limited, 2001. – 1034 p.

За контакти:
Маг. инж. Драгомир Христов Славов
Фирма “Цифрови системи” – Варна

Милена Янкова Караиванова
Специалност “Компютърни системи и технологии”
Технически университет – Варна

доц. д-р инж. Петър Цветанов Антонов
Катедра “Компютърни науки и технологии”
Технически университет – Варна
E-mail:peter.antonov@ieee.org

Software Solutions in Infrared Thermography

Geo V. Kunev

Abstract: This paper describes briefly the usage of theory and practices both in infrared (IR) technology and in IT technologies in experimental and application work on the some of basic IT - IR thermography tasks. Based on experimental IR images and IT surveys are developed two software systems:

- IR Image sharpness improvement system
- IR video analog to digital transferring system

Софтуерни решения в инфрачервената термография

Гео В. Кунев

Резюме: В доклада се описват накратко използването на теория и практика в инфрачервената (IR) технология и IT технологиите за експериментална и приложна работа в някои от базовите IT – IR термографски задачи. На основата на експериментални IR изображения и ИТ изследвания са разработени две софтуерни системи:

- Система за подобряване на IR изображенията
- Система за трансфер на IR аналогово видео в цифрова форма.

1. Introduction

Infrared is an invisible portion of the light spectrum extending from 0.75 to 1000 microns. All objects warmer than absolute zero (0 Kelvin or -275.15°C) emit energy somewhere within that range [1].

The one of the problems there is low image resolution of IR cameras. The best IR cameras have image resolution of about only 320 x 240 pixels.

Even worse is the situation with older cameras, like the IR thermo camera Agema 470 with image resolution of 140 x 140 pixels. Because of thermo-noise and usage of specific singular low temperature detectors, that use rotating mirror system, IR images are not sharp.

The first goal of this task is to reach the better IR camera image quality with software sharpening, specific to the IR images and camera characteristics. The second goal is to convert analog IR cameras video output to digital image form.

2. Infrared Image Sharpening

A. Experimental IR Images

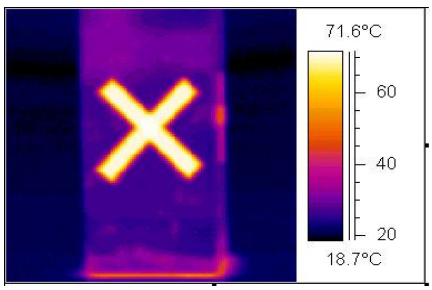


Fig. 1. An experimental IR image

The experimental IR images show that edges has no sharp outline.

Sharpening the whole image using image sharpening algorithms is dangerous to the exact values of brightness that hold substantial information about IR radiation of examined objects and possibility to calculate temperatures.

B. Edge sharpening solution

A possible solution is to process only edge zone, keeping values of pixel brightness, using approaches, similar to [2].

As can be seen on Fig.1 an IR image edge has platform shape (shown on Fig. 2). If we can make this platform less oblique, but do not change the other parts of the image, thus we will sharpen the edges keeping exact values of pixel brightness (Fig. 3).

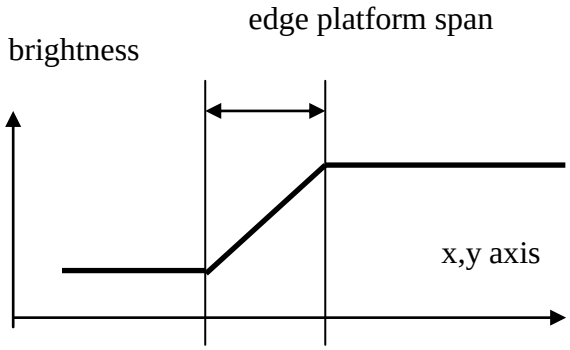


Fig. 2. An IR image edge before processing

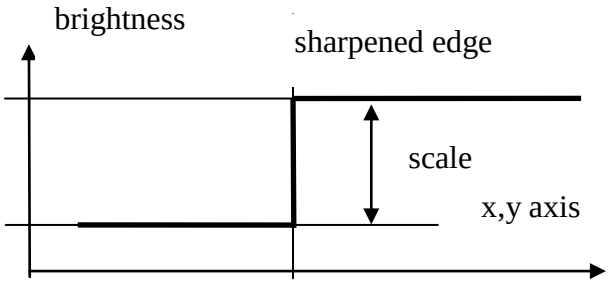


Fig. 3. An IR image edge after processing

C. IR Image Edge Sharpening Software

A C++ software system (Fig. 4) was developed, offering two kinds of algorithms:

- an algorithm used in computer tomography.
- Geo Kunev algorithm based on above solution.

Both work with 140 x 140 pixel images of IR camera Agema 470, making images more clear, sharpening the edges of objects.

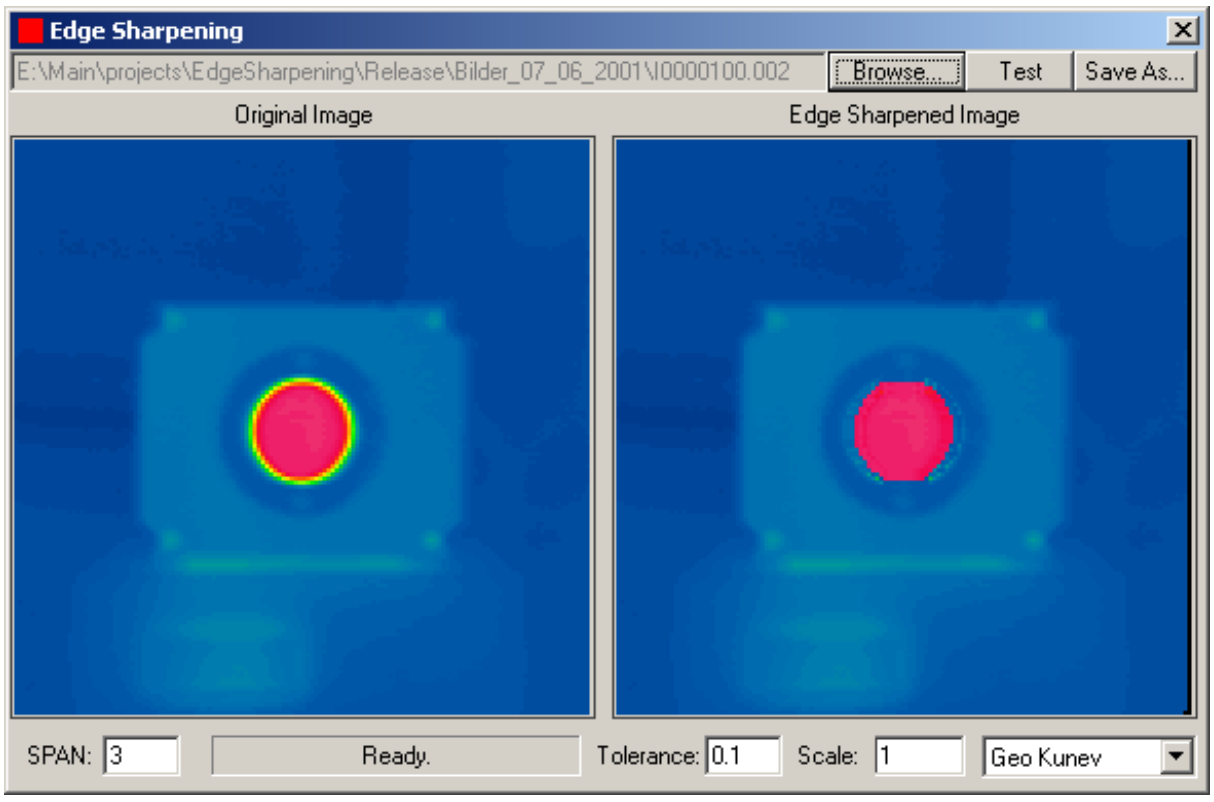


Fig. 4. The IR image edge sharpening software

Using the second algorithm, the software search edges with span, scale and tolerance, set by user then process the image, sharpening the edges in the manner, shown in Fig. 3.

A software system was tested with both algorithms, and with different spans, scales and tolerances.

D. IR Image Edge Sharpening Code

Samples of C++ (VS) code:

```
…
CEdgeSharpeningDlg Nachrichten-Handler
BOOL CEdgeSharpeningDlg::OnInitDialog()
{  Dialog::OnInitDialog();
   ASSERT((IDM_ABOUTBOX & 0xFFF0)
== IDM_ABOUTBOX);
  ASSERT(IDM_ABOUTBOX < 0xF000);
  CMenu* pSysMenu =
GetSystemMenu(FALSE);
   if (pSysMenu != NULL)
  {  CString strAboutMenu;

strAboutMenu.LoadString(IDS_ABOUTBO
X);
          if (!strAboutMenu.IsEmpty())
           { pSysMenu-
>AppendMenu(MF_SEPARATOR);
            pSysMenu-
>AppendMenu(MF_STRING,
IDM_ABOUTBOX, strAboutMenu);
          }
  }
  SetIcon(m_hIcon, TRUE);
  SetIcon(m_hIcon, FALSE);
  int iItem;
  iItem = m_Type.InsertString(0, _T("Geo
Kunev"));
  m_Type.SetItemData(iItem, 1);
  m_Type.SetCurSel(iItem);
  iItem = m_Type.InsertString(1,
_T("Tomography"));
  m_Type.SetItemData(iItem, 2);
  m_Type.SetItemData(iItem, 3);

m_FileName.SetWindowText(_T("C:\\Eigene
Dateien\\I0000100.003"));
          MakePalette();
          Load_IR_Image();
          return TRUE;
  }
…
BOOL
CEdgeSharpeningDlg::ProcessImage2(unsign
ed char *pImage)
{  double tol = m_dTolerance, scale =
m_dScale;
  int iRows = IMAGE_HEIGHT, iColumns =
IMAGE_WIDTH, SPAN = m_iSpan, k, m, x,
y;
  double  InputImage
[IMAGE_WIDTH][IMAGE_HEIGHT];
  double T1[IMAGE_WIDTH]
[IMAGE_HEIGHT];
  double
T2[IMAGE_WIDTH][IMAGE_HEIGHT];
  double
Q2[IMAGE_WIDTH+4][IMAGE_HEIGHT];
  double
T3[IMAGE_WIDTH+4][IMAGE_HEIGHT];
  double
TZ[IMAGE_WIDTH+4][IMAGE_HEIGHT];
  double
T4[IMAGE_WIDTH][IMAGE_HEIGHT];
  double
TY[IMAGE_WIDTH][IMAGE_HEIGHT];
  double
T5[IMAGE_WIDTH][IMAGE_HEIGHT];
  int i=0;
  for(i=0; i<2; i++)
  {
   if(i==0)
    {
// HAAR WAVELET TRANSFORM =>
VERTICAL LINES
     for(k=0; k<iRows; k++)
       for(m=0; m<iColumns; m++)
         InputImage[m][k] =
(double)pImage[k*iRows + m];
      }
     else
      {
// HAAR WAVELET TRANSFORM =>
HORIZONTAL LINES
   for(k=0; k<iRows; k++)
    for(m=0; m<iColumns; m++)
       InputImage[m][k] = T5[k][m];
        }
   for(k=0; k<iRows; k++)
        {
         for(m=0; m<iColumns-1; m++)
           {T1[m][k] = scale *
(InputImage[m][k] + InputImage[m+1][k]);
             T2[m][k] = scale * (-
InputImage[m][k] + InputImage[m+1][k]);
```

```
            }
          T1[m][k] = scale * InputImage[m][k];
          T2[m][k] = scale * (-
InputImage[m][k]);
             }
   for(k=0; k<iRows; k++)
     for(m=0; m<iColumns; m++)
          Q2[m][k] = 0.0;
          double dMaximum =
FindMaximum((double *)T2, iRows *
iColumns);
          for(k=0; k<iRows; k++)
            for(m=0; m<iColumns; m++)
             Q2[m+2][k] =
fabs(T2[m][k])/dMaximum;
            for(k=0; k<iRows; k++)
                  for(m=0; m<iColumns+4; m++)
                       T3[m][k] = 0.0;
                 for(k=0; k<iRows; k++)
                   for(m=0; m<iColumns; m++)
                        T3[m+2][k] = T2[m][k];
…
// INVERSE HAAR WAVELET
TRANSFORM
for(k=0; k<iRows; k++)
  for(m=0; m<iColumns; m++)
T5[m][k] = (scale * (TY[m][k] -
T4[m][k]))/2.0;
}
…
```

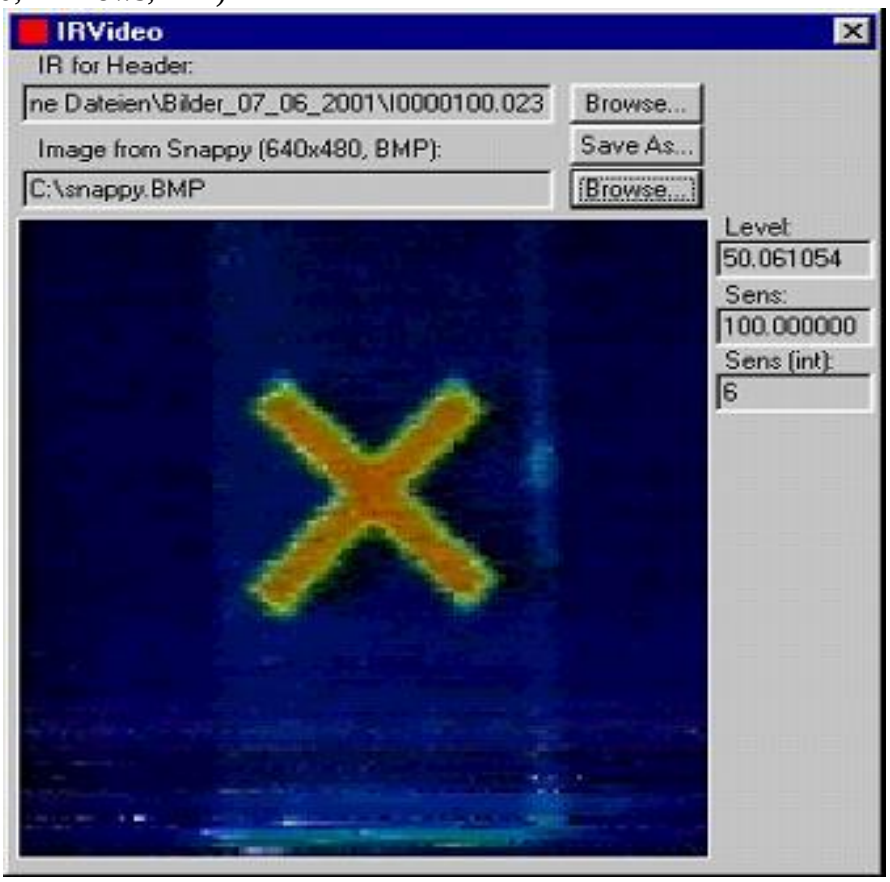


Fig. 5. IR video – digital transferring software

3. IR Video – digital Transferring System

IR cameras save static IR images in digital form, making possible further processing, including temperature calculating. They have also real-time analog video output.

The developed (C++) IR video – digital transferring system (Fig. 5) transfer captured and saved to a file analog video output to the standard static digital IR image form, using initial IR digital image as a header, giving IR camera parameters.

4. Conclusion

Both systems where tested and successfully used in assistance of the IR thermography investigations [3].

References

[1] B. Breukmann, Kapitel Wärmebildkameras und IR-Bildverarbeitung und optische Messtechnik, München,1993

[2] M.Stoeva, V.Bojikova, Image Similarity on Relative Spatial Location of Image Objects in SDI Databases, V ICCS for Design and Technology, pp. 224- 226, Moskva, Russia, 5-7nd October, 2005

[3] V.Naumov, G.Kunev, R.Arsenov, On data problems in technical and non-technical applied systems using infrared thermography models, Механика на машините, книга 4, 2002

За контакти:
гл.ас. д-р Гео В. Кунев
катедра Компютърни науки и технологии
Технически университет – Варна
E-mail: geo_qnew@hotmail.com

ГЕНЕРИРАНЕ НА ОСВЕТЕНОСТ НА КОНТУРНИ ИЗОБРАЖЕНИЯ ПОСРЕДСТВОМ АВТОВЪЛНОВ ПРОЦЕС

Огнян И. Железов

Резюме: В тази статия се предлага алгоритъм за генериране на осветеност на контурни изображения посредством автовълнов процес (АВП) в дискретна среда. Използва се модификация на бинарният АВП, като растерното изображение се представя като мрежа от дискретни елементи, всеки от които може да бъде в едно от 256^3 възможни състояния. Предлаганият модел на АВП реализира еднопосочно разпространение на състоянието на осветеност по зададен закон на изменение на яркостта на точките. Като резултат се получава изменение на яркостта на точките от вътрешността на контурните обекти, пропорционално на разстоянието от контура на обекта. Алгоритъмът притежава предимствата, характерни за алгоритмите, използващи АВП – високо бързодействие и възможност за паралелно изпълнение на операциите.

Lighting Objects Using Autowave Propagation

Ognyan I. Zhelezov

Abstract: In this paper is proposed one algorithm for lighting of contour images using autowave propagation (AWP) in excitable media. Algorithm uses one modification of discrete AWP. Source rastered image is described as a net of discrete elements. Propagation of AWP within this media is a process of switching of elements from state 255 (white) to state, obtained from the brightness of neighbor elements, having color, different from white. As a result, the value of pixel's brightness inside an objects contour is changed as a function of distance from pixel (AWP element) to contour curve. Algorithm has the advantages of AWP algorithms – fast performing and possibility parallel execution.
Key words: autowaves propagation, lighting of objects

1. Увод

Генерирането на осветеност на двумерни изображения с цел да се получи илюзия за тримерност, по правило се извършва в интерактивен режим. Графичните 3D редактори предоставят функции, чрез които може да се преобразуват двумерни изображения в определени видове тримерни изображения и като резултат да се получи тяхното фотореалистично осветяване. В статията се предлага алгоритъм за генериране на осветяване на двумерни контурни изображения посредством автовълнов процес (АВП), който се изпълнява с по-малък брой операции при определени ограничения на функцията на изменение на яркостта и на положението на източника на осветяване

2. Общи сведения за автовълновите процеси

Автовълновите процеси (АВП) са процеси на разпространение на нелинейни вълни във възбудима среда. Те се описват посредством нелинейни диференциални уравнения от вида: [1]

$$\frac{du}{dt} = D.\frac{d^2u}{dx^2} + f(u) \qquad (1)$$

Съществуват различни модели на разпространение на АВП в дискретна среда.При най-простия бинарен модел всеки елемент от дискретната среда има само две възможни състояния (можем да ги обозначиш условно с нула и единица). Разпространението на вълната при бинарен

АВП се извършва по следния алгоритъм:
Д1)Ако за даден такт от дискретното време един елемент е в състояние 0 и има поне един съседен елемент, който е в състояние 1, то по време на следващия такт той превключва в състояние 1. В противен случай елементът запазва състоянието си. Дори този най-прост модел на АВП дава възможност за извършване на ефективна паралелна обработка на бинарни изображения – възстановяване на прекъснати контури, изглаждане на контури, скелетизация и др. В доклада се предлага алгоритъм за генериране на осветеност на контурни изображения посредством АВП в дискретна среда с 256^3 възможни състояния на елементите, съответстващи на 8-битово кодиране на яркостта на трите основни цвята на точките от изображението.

3. Формулировка на задачата за генериране на осветеност.

Генерирането на двумерни изображения със зададена форма по правило започва с получаването на контурно изображение – изображение, което съдържа само контурни криви, които отделят обекта от фона на изображението. Генерирането на осветеност на обект, представен с контурна крива създава излюзия за тримерност, което подобрява вида на изображението.
Ако се приеме, че контурната крива на обекта е получена от двумерна проекция на тримерен обект, то очевидно тя не съдържа достатъчно информация, по която да може да бъде възстановен тримерният обект. Известната в теорията на цифровата обработка на изображения теорема за *проекционния срез* доказва, че едно тримерно изображение може да бъде възстановено напълно чрез определен брой проекции, получени за проекционни ъгли, равномерно разположени в интервала от 0 до 2π. В случай, че разполагаме само с контурно изображение, получено от една двумерна проекция, можем чрез него да получим тримерен обект (и съответно осветеност на обекта) само ако направим определени предположения за формата на обекта. Съвременните програмни средства за тримерна графика (3DMAX, OpenGL, DirectX) предлагат две основни функции за преобразуване на контурно изображение в тримерен обект – изтегляне (екструдер) и ротация. Екструдерът преобразува контурната крива в криволинеен цилиндър, ограничен от контурната крива и сечението на две успоредни равнини. Ротацията на контурната крива около избрана ос създава тримерен обект с кръгова симетрия. Полученият тримерен обект се описва чрез координатите на върховете на мрежа от полигони, като осветяването на всеки полигон се изчислява чрез осветеността на неговите върхове. Тези алгоритми за генериране на тримерни обекти и осветяването им, въпреки че гарантират получаването на фотореалистични обекти със зададени геометрични характеристики и осветеност, изискват изпълнението на голям брой изчислителни опреации. В някои случаи, например при генерирането на тримерни текстури, могат да се поставят ограничени изисквания за точност на апроксимацията и да се използва само определен вид осветяване, но изискват повишено бързодействие. Предлаганият алгоритъм за генериране на осветеност на контурни изображения използва автовълнов процес (АВП) в дискретна среда и притежава предимствата, характерни за алгоритмите, използващи АВП – високо бързодействие и възможност за паралелно изпълнение на операциите. Ограниченията на алгоритъма са във вида на градиента на яркостта (само в посока, препендикулярна на контурната крива и използването само на определен вид осветяване от направлението на наблюдателя.
Формулираме задачата за генериране на осветяване на контурни обекти по следния начин:

1. Зададено е изображение, което съдържа бял фон и обекти, ограничени със затворени контурни криви и запълнени също с бял цвят.
2. Зададена е функцията на осветяване

Fb(n), , която изразява яркостта точка от обекта като функция на разстоянието **n** от точката до контурната крива.

3. Генерирането на осветяване на обект, ограничен от затворена контурна крива означава да се получи яркостта на всяка точка от вътрешността на обекта като функция Fb(n), на разстоянието n от точката до контурната крива.

4. Модел на АВП с 256 състояния на елементите.

В описания в т. 2 алгоритъм за разпространение на АВП в бинарна дискретна среда се определя условието за преминаване на един елемент от състояние 0 в състояние 1 като проста логическа функция от състоянието на съседните елементи. За генериране на осветеност контурни обекти в дискретно изображение се предлага да се използва модел на АВП с брой състояния на дискретните елементи, равен на броя на нивата на яркост, с които се кодира дискретното изображение. В използваните графични файлови формати най-големият брой нива на яркост, с който се кодират трите основни цвята е 256 и съответно цветът на точка от изображението се задава с функцията RGB(red, green, blue), където с red, green и blue се задава яркостта (от 0 до 255) на трите основни цвята. В модела на АВП на точките с бял цвят съответстват активни елементи, а на всички останали точки (с цвят, различен от бял) съответстват пасивни елементи. При всяка стъпка от АВП активните елементи променят състоянието си само ако имат поне един съседен пасивен елемент. Състоянието, в което преминава активният елемент се определя от зададената функция на осветяване Fb(n) и номера n на стъпката от изпълнение на АВП. В [1] състоянието на елемент, което не се променя при изпълнение на АВП независимо от състоянията на съседните елементи се нарича пасивно състояние. В описаният модел на АВП само състоянието RGB(255,255,255). е активно, а всички останали състояния са пасивни.

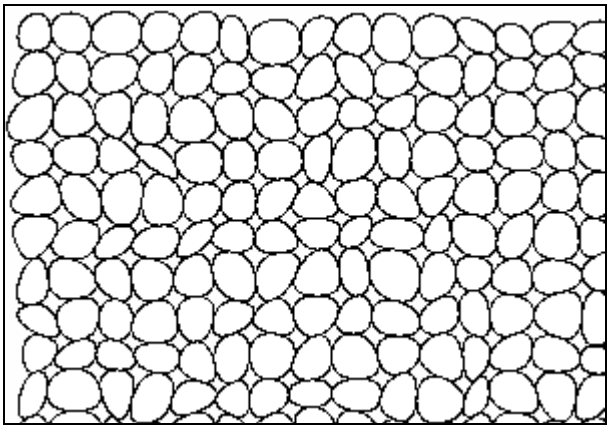
5. Описание на алгоритъма

Алгоритъмът включва следната последователност от операции:

1. Запълва се фонът на зададеното изображение с цвят, съответстващ на пасивно състояние, например RGB(254,254,254).
2. Изпълняват се N стъпки на АВП. В съответствие с алгоритъма на АВП на n-тата стъпка от АВП ако една точка (дискретен елемент) от изображението има бял цвят и има поне една съседна точка, която не е с бял цвят, то точката получава цвят Fb(n). Ако всички съседни точки са с бял цвят, то дадената точка не променя цвета си.

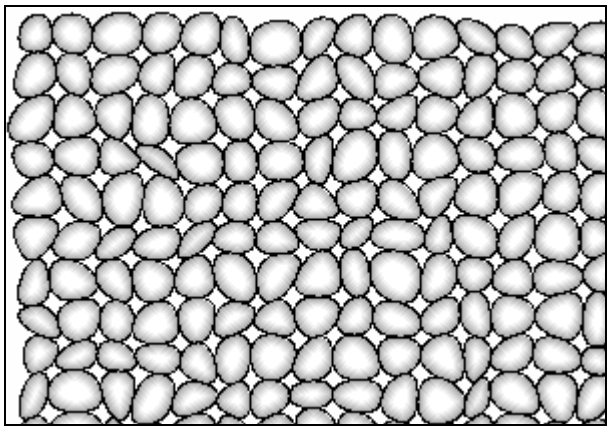
6. Практически резултати.

Пример за приложение на описания алгоритъм е показан на фиг. 1 и фиг. 2. Първоначалното изображение е показано на фиг. 1. То представлява текстура, получена чрез паркетиране на правоъгълник със случайни четириъгълници и последващо изглаждане на контурите на четириъгълниците.



Фиг. 1 Зададено контурно изображение

Резултатът от изпълнението на описания алгоритъм за осветяване при 8 стъпки на АВП (N=8) е показан на Фиг. 2



Фиг. 2 Резултат от прилагането на алгоритъма за осветяване на контурни обекти към изображението от Фиг. 1

От резултата се вижда, че генериране на осветеност на контурните обекти в зададеното изображение създава илюзия за тримерност и значително подобрява вида на текстурата.

7. Заключение

Предлаганият алгоритъм дава възможност генериране на осветеност на обекти в контурни изображения. Посредстваом изменение на яркостта на точките от вътрешността на контурните обекти, пропорционално на разстоянието от контура на обекта. Алгоритъмът притежава предимствата, характерни за автовълновите алгоритми – локалност, рекурсивност и възможност за паралелно изпълнение. За метода е характерно също така, че той не изисква предварителен анализ на формата на контурите, за които ще се генерира осветеност. Като недостатък на алгоритъма може да се посочи това, че той може да се прилага само за контурни изображения и само за светлинен източник, разположен в направлението на наблюдателя.

8. Литература

[4] .В. С. Зыков, Моделирование волновых процессов в возбудимых средах. М. “Наука” 1984г.

[5] О. Железов, Възстановяване на контурни изображения посредством автовълнов процес, списание “Автоматика и информатика”, брой 5-6, 1998г

[6] О. Железов, Генериране на осветеност на обекти посредством автовълнов процес, сборник доклади от конференция “Морски научен форум”, том 2, 1998г

[7] Л. Рабинер, Б. Гоулд, Теория и применение цифровой обработки сигналов. М. “Мир” 1978г.

[8] Kung s, Whitehouse H and Kailath editors, VLSI and modern signal processing, Prentice hall inc. New Jersey, 1989

[9] Т. Павлидис, Алгоритмы машинной графики и обработки изображений. пер. Н.Гуревич, М. “Радио и связь”, 1986г.

[10] D. E. Dudgeon and R. M. Mersereau, Multidimensional signal processing, Prentice Hall, Inc. 1984.

За контакти:
доц.д-р инж. Огнян И. Железов
катедра “Компютърни науки и технологии”
Технически университет - Варна
E-mail:ognyanz@yahoo.com

АЛГОРИТЪМ ЗА ПРОСЛЕДЯВАНЕ НА КОНТУРА НА ОБЕКТ СЪС СЛЕДЯЩА ТОЧКА, ДВИЖЕЩА СЕ В ОБЛАСТТА НА ФОНА НА ИЗОБРАЖЕНИЕТО

Огнян И. Железов

Резюме: В тази статия се предлага алгоритъм за проследяване на контура на обекти в бинарно растерно изображение. Алгоритъмът се различава от извстните алгоритми за проследяване по това, че използва следяща точка, която се движи по границата между обекта и фона, но като остава винаги в областта на фона на изображението. Това дава възможност някои недостатъци на известните алгоритми за проследяване. Алгоритъмът позволява използването на най-простият критерий за край на проследяването без това да води до пропускане на сканирането на части от контура. Алгорътъмът и изисква по-малък брой операции в сравнение с известните алгоритми за проследяване на обекти с осмична връзка на елементите, тъй като проверява само 4-свързаните елементи

Contour Tracing Algorithm Using a Following Point Outside the Pattern.

Ognyan I. Zhelezov

Abstract: This paper proposes a fast and high-definition contour-tracing algorithm for digitized binary images, which avoid the weakness of known algorithms having less number of operations for one step of tracing. The known algorithms like Square Tracing algorithm, Moore-Neighbor Tracing and others uses following contour point, which is moved along the pattern boundary, but remaining always inside the pattern. The proposed algorithm uses following contour point, remaining every time outside the pattern, which makes the stopping criterion more simple and reliable.

Key words: contour following, contour tracing

1. Увод

Проследяването на контура е една от полезните предварителни обработки на изображения. Контурната крива дава възможност за извличане на различни признаци за формата на обекта, които се използват за класификацията на обекти и за анализ на сцени.

2. Бинарни изображения. Формулиране на задачата за проследяване на контура.

Бинарното изображение може да се определи като мрежа от дискретни елементи, всеки от които може да бъде в едно от следните състояния:

Състояние 1 – ако елементът принадлежи на обект от изпбражението.

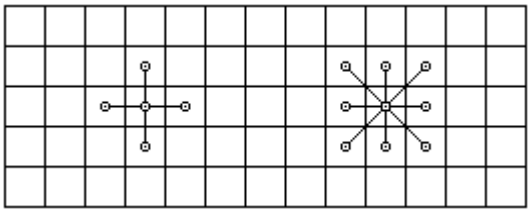
Състояние 0 – ако елементът принадлежи на фона изпбражението

Графичният файлов формат, който най-точно представя бинарните изображения е monochrome bitmap, който представя изображението като матрица от точки с бял или черен цвят. За по-нататъшните разглеждания ще считаме, че точките с бял цвят принадлежат на фона на изображението, а точките с черен цвят принадлежат на обект от изображението.

Обектът в бинарното изображения при тази кодировка на състоянието на елементите може да се определи като област от свързани черни точки. Две точки

се считат за свързани, ако между тях съществува поне една непрекъсната последователност от съседни точки.

Дадената дефиниция на свързаност на точки в бинарно изображение изисква определение на това, кои точки се считат за съседни. Отговорът на този въпрос е свързан с дефинирането на съседство на елементи в правоъгълна мрежа. Дадената по-долу фигура показва две възможни определения за съседство.



Фиг. 1 Четворна и осмична връзка между елементи от правоъгълна мрежа

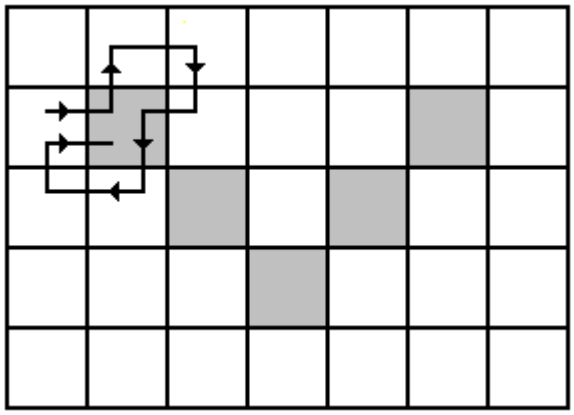
4-свързаност (четворна връзка) – за съседни се считат елементите, които имат обща страна с дадения елемент. В правоъгълна мрежа с четворна връзка всеки елемент има четири съседни елемента.

8- свързаност (осмична връзка) – за съседни се считат елементите, които имат обща страна или общ връх с дадения елемент. В правоъгълна мрежа с осмична връзка всеки елемент има осем съседни елемента.

Счита се [2], че алгоритмите за проследяване на контура трябва да могат да извършват проследяване обекти с 8-свързаност на елементите.

3. Проблемът с критерия за край на проследяването.

Най-старият известен алгоритъм за проследяване на контура на обекти в дискретно изображение е XY алгоритъмът (Square Tracing Algorithm). Проследяването започва от начална точка, която принадлежи на контура на обекта. За проследяване на контура е необходимо да се изпълни следната последователност от стъпки:



Фиг. 2 Пример за проследяване на контура на обект с осмична връзка по XY алгоритъм (Square Tracing Algorithm)

- когато текущият пиксел е с черен цвят (т.е. принадлежи на обекта) се прави завой наляво.
- когато текущият пиксел е с бял цвят (т.е. принадлежи на фона) се прави завой надясно.

Тези стъпки се изпълняват докато следящата точка достигне отново началната точка, от която е започнало сканирането.

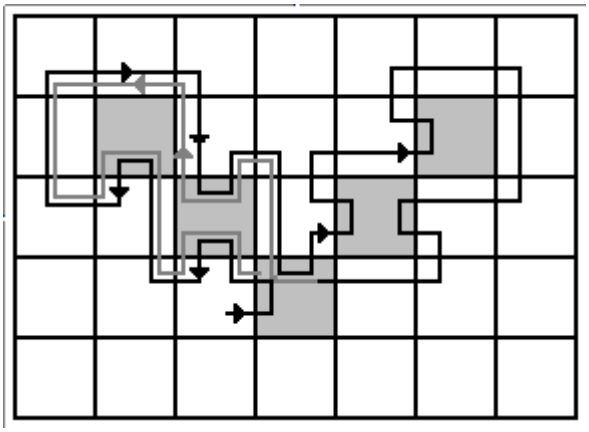
Основният недостатък на XY алгоритъмът е това, че той не може да се използва за проследяване на обекти с осмична връзка между елементите. На дадената по-горе фигура е показан пример за изпълнението на алгоритъма при проследяване по контура на обект с осмична връзка. Вижда се, че алгоритъмът не успява да открие следващичт пиксел от обекта и завършва проследяването още на първият пиксел.

Друг известен аглоритъм, който дава възможност за проследяване по контура и на обекти с осмична връзка е алгоритъмът Moore-Neighbor Tracing – При всяка стъпка от проследяването на контура се изпълняват следните операции:

- когато текущият пиксел е с черен цвят (т.е. принадлежи на обекта), следящата точка се връща обратно на белият (т.е. принадлежащ на фона) пиксел, на който е била преди това, и започва да проверява съседните (по осмична връзка)

точки по посока на часовниковата стрелка, докато открие черен пиксел, на който да се премести.

Такива стъпки се изпълняват докато следящата точка достигне отново началния пиксел от контура, от който е започнало проследяването..



Фиг. 3 Пример за проследяване на контура на обект с осмична връзка по алгоритъм Moore-Neighbor Tracing

Примерът, наден по-горе (Фиг. 5) показва движението на следящата точка при проследяване на същия обект, както в предния пример, но при условие, че начален е най-долния пиксел. Примерът показва основният недостатък на алгоритъма Moore-Neighbor Tracing – необходимостта от по-сложен критерий за край на проследяването в сравнение с XY алгоритъма. Критерий за край на проследяването, който гарантира завършване без пропускане на части от контура е предложен от Jacob Eliosoff: “Проследяването приключва, когато следяшата точка навлезе отново в началния пиксел, от който е започнало проследяването, по същия начин (което означава “от същата посока”). Примерът, показан на Фиг. 5 показва че следящата точка ще навлезе в пиксела, от който е започнало проследяването по същия начин, както в началото, след един пълен цикъл около контура на обекта и един допълнителен цикъл около лявата част от изображението. (следящата линия на втория цикъл е показана със сив цвят).

Известни са и други алгоритми за проследяване по контура [2],[3],[4], например: радиално сканиращ алгоритъм (Radial Sweep) и алгоритъмът, предложен от Theo Pavlidis, но за тях също е необходим усложнен алгоритъм за край на проследяването, в резултат на прилагането на който е необходимо допълнително сканиране на част от контура. В настоящата статия се предлага алгоритъм за проследяване на контура на обекти, при който се гарантира завършване на проследяването достигане до началната точка без да е необходимо допълнително проследяване на части от контура.

4. Описание на предлагания алгоритъм за проследяване на контура на обекти.

По предлагания алгоритъм движението на следащата точка наподобява движението на човек, който заобикаля препятствие, като при всяка крачка докосва препятствието с дясната си ръка. (Това е вариант на известния алгоритъм за обхождане на лабиринт). Следящата точка се движи винаги в областта на фона по границата между обекта и фона. Съседните пиксели на текущия пиксел се проверяват в посока, обратна на часовниковата стрелка, като пеоверката започва от пиксела, който е вдясно от последното преместване на следящата точка. Изключение се прави за проверката на съседните пиксели на **start_pixel** – началната точка, от която започва проследяването. При всяка стъпка на алгоритъма координатите на откритите черни пиксели се записват без повторения в масив като пиксели от контура на обекта. Освен това пикселите, през които преминава следящаъта точка могат да се запишат като пиксели на вуншния контур на обекта. Контурът на обекта се проследява по посока на часовниковата стрелка.

Промяната на последователността на проверка на съседните точки на обратна (по часовниковата стрелка) и промяната на началният пиксел, от който започва

проверката на левия спрямо последното преместване на следящата точка (десен за началната точка на проследяване) ще има за резултат проследяване на контура по посока, обратна на часовниковата стрелка.

По-долу е дадено формално описание на предлагания алгоритъм като последователност от стъпки. Текстът след "//" е коментар):

Input: е правоъгълна мрежа, **T**, съдържаща свързан компонент (обект) **P** от черни клетки.

Output: е последователност (масив) **border** (b1, b2 ,..., bk) от гранични пиксели, които формират контура на обекта.

current е пикселът, на който текущо е позиционирана следящата точка.

checked е пикселът, който текущо се проверява при проверката на съседните (по четворна връзка) пиксели на **current.**

directions е масив, който съдържа последователността от възможни премествания на следящата точка {"RIGHT","DOWN","LEFT","UP"},

Възможните прместванията са в положително или отрицателно напревление на Декартовите координатни оси. Достъпът до елементите на масива е последователен като при четене от цикличен буфер, което означава например, че елементът "UP" е следващият след "RIGHT" при четене в посока, обратна на часовниковата стрелка.

direction_of_checking е посоката от текущият пиксел **current** pixel to проверяваният пиксел **checked.**

direction_of_movement е посоката от предходната позиция на текущият пиксел **current** до текущата му позиция **.**

Begin

1 Изчиства се масивът **border**.
2 Търси се точка от изображението, от която да започне проследяването на контура. Това може да се направи по различен начин, например като се сканира изображението отляво надясно и отдолу нагоре по редове до откриването на черна точка.
3 Откритият черен пиксел се записва като **checked**.
4 Записва се пикселът **checked** в масива **border**.
5 Записва се предходният бял (т.е. от фона) пиксел като **current** и се записва като **start_pixel** – начална точка на проследяването.
6 Записва се посоката на последното преместване от пиксела **current** към пиксела **checked** като **direction_of_movement.**
7 Прави се завъртане наляво на посоката на движение - записва се в **direction_of_ movement** следващата обратна на часовниковата стрелка посока на движение от масива **directions.**
8 Записва се в **direction_of_checking** следващата по часовниковата стрелка посока от масива **directions** спрямо посоката да движение.
9 Записва се в **checked** пикселът, съседен на **current** по направление на **direction_of_checking.**
10 Проверява се цветът на пиксела **checked.**

a Ако цветът на **checked** е черен:
// пикселът checked принадлежи на:
// обекта - запис в **border** и завъртане
//наляво

- Записва се в **direction_of_checking** следващата по направление, обратно на часовниковата стрелка посока от масива **directions** //Завъртане наляво
- Дабавя **current** към **border** ако не съвпада с последният добавен пиксел.

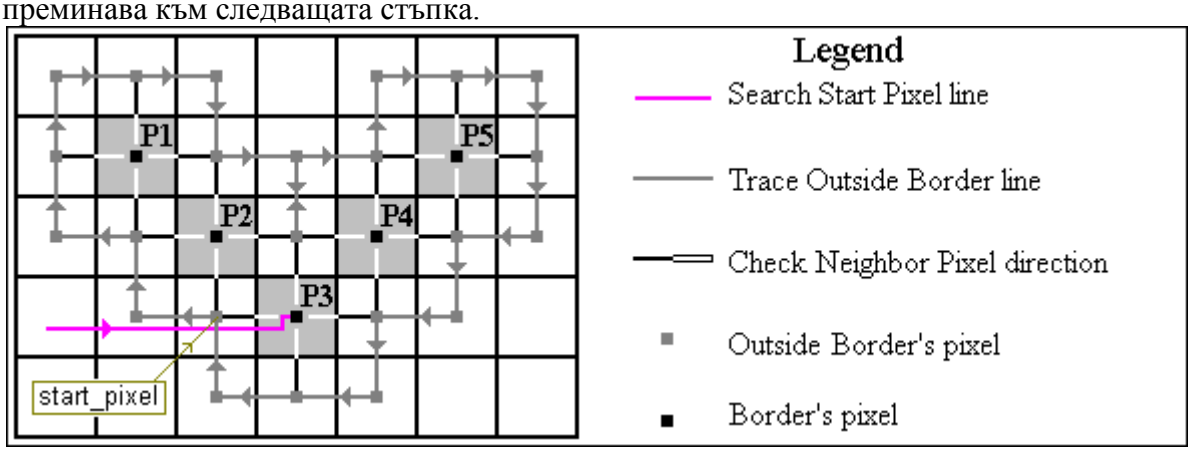
// Запис

- Преминава се към стъпка 9.

b Ако цветът на **checked** е бял:
// пикселът checked принадлежи на:
// фона - запис в **border**, преместване
// в тази точка или КРАЙ

- Записва се в **direction_of_movement** посоката от пиксел **current** към пиксел **checked.**
- Записва се в **current** последната стойностт на **checked.**

- Проверява се дали новата стойност на **current** съвпада със **start_pixel**. Ако съвпада, КРАЙ , Ако не съвпада се преминава към следващата стъпка.
- Преминава се към стъпка 8.



Фиг. 4 Пример за проследяване на контура на обект с осмична връзка по предлагаиня алгоритъм

Дадената по-горе фигура показва разултата от изпълнението на предлагания алгоритъм за изображението от Фиг. 3.

Търсенето на пиксел от обекта започва от най-долния ляв елемент. Първия черен пиксел, който ще бъде открит, е P3. Пиксел P3 ще бъде записан в масива **border** като пиксел от контура. Предходният ляв пиксел ще бъде записан като **current** и като **start_pixel**. Посоката на движение, която ще се запише след завъртане наляво в **direction_of_movement** ще бъде "UP". Стойността на **direction_of_checking** ще бъде следващата по посока на часовниковата стрелка от масива **directions,** следователно "RIGHT". Тъй като това е пиксел от обекта, посоката за проверка **direction_of_checking** ще бъде променена на следващата в посока, обратна на часовниковата стрелка, т.е. на. "UP". Проверката на пиксела над current показва, че той също е от обекта, и следователно той се записва в масива **border** като пиксел от контура. Следващият **checked** пиксел ще бъде в посока "LEFT". Тъй като това е пиксел от фона, той се записва като текущ пиксел current и в **direction_of_movement** се записва "LEFT",.

Това е изпълнението на една стъпка от проследяването на контура на обекта. За следващата стъпка първият проверяван пиксел ще бъде в посока "UP" (следващата по посока на часовниковата стрелка). Тъй като проверяваният (**checked)** пиксел е от фона, следящата точка ще се премести на тази позиция (пикселът ще се запише като **current**). … Проследяването ще продължи докато пикселът **current** достигне до началната точка на проследяване **start_pixel**. След завършване на проследяването масивът **border** ще съдържа координатите на следната последователност от пиксели {P3. P2, P1, P2, P3, P4, P5, P4, P3}. Нека да припомним, , че пикселите се записват в масива без повторения. Например пикселът P2 се проверява два пъти последователно, нп ще бъде записан в масива **border** само веднъж.

5. Заключение

Предлаганият алгоритъм има някои предимства в сравнение с представените в [3] алгоритми за проследяване по контура. Алгоритъмът използва най-простият критерий за завършване на проследяването – достигане до началната точка, и винаги

се изпълнява за целия обект. За предлаганият алгоритъм достигането до началната точка **start_point** означава пълен цикъл около границата на обекта. Друго предимство на алгоритъма е това, че той дава възможност за получаване не само на контура на обекта, но и на външния (обвиващ) контур на обекта, който се състои от пикселите от фона, по които се движи следящата точка.

Литература

[1] Duda R., Hart P. Pattern Classification and Scene Analysis, Standford Recearch Institute, Menlo Park, California 1973.

[2] T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, Maryland, 1982.

[3] Abeer George Ghuneim, http://www.cs.mcgill.ca/~aghnei/index.html., Contour Tracing.

[4] Л. Рабинер, Б. Гоулд, Теория и применение цифровой обработки сигналов. М. “Мир” 1978

За контакти:
доц.д-р инж. Огнян Иванов Железов
катедра “Компютърни науки и технологии”
Технически университет - Варна
E-mail:ognyanz@yahoo.com

КОНВЕЙЕРЕН УМНОЖИТЕЛ С КОНЦЕНТРАТОРИ

Димитър С. Тянев, Стефка И. Попова, Драгомир В. Янев, Александър И. Иванов

Резюме: Представено е едно ново приложение на концентратори от тип 3:1 в схеми на конвейерни умножители, с помощта на които скоростта се увеличава двойно. Показано е, че приложението на концентраторите от този тип, е най-доброто в сравнение с известните алгоритми за умножение на 2 разряда едновременно. Представена е оригинална принципна логическа схема на комбинационната част за всяко ниво в умножителя, както и общата логическа структура на конвейерния умножител. Работоспособността и качествата на предлаганата схема за построяване на конвейерни умножители са експериментирани и показани върху продукти на фирма Xilinx.

Conveyer multiplier with concentrators

Dimitar S. Tyanev, Stefka I. Popova, Aleksandar I. Ivanov, Dragomir V. Yanev

Abstract: A new application of 3:1 concentrators in schemes of conveyer multipliers are introduced to double acceleration of the speed. It is showed that the application of such concentrators is the best in comparison with well-known algorithm for two bits together multiplication. An original logical scheme of the combinational part for each level in the multiplier is introduced, as well as the general logical stucture of the conveyer multiplier. The efficiency and qualities of the proposed scheme for building conveyer multipliers are tested and showed through Xilinx products.

1. Въведение

Операция умножение на цели числа е с висока честота на изпълнение в компютърните изчисления. Съвременните цифрови процесори се характеризират с това, че тяхната операционна част е конвейерно организирана. Комбинационните схемни умножители, въпреки изключително прецизния си логически синтез [1÷9], предвид голямата дължина на операндите, внасят закъснение, което е съществено по-голямо от периода за тактуване на командния конвейер. Ето защо тук е разгледана като актуална конвейерната организация на схемния умножител [13÷16].

2. Същност на конвейерния умножител

Възможната логическа структура на конвейерния умножител е представена подробно в [10]. Тази структура се характеризира с това, че съдържа n на брой нива (фиксатори) за едновременно движещите се междинни суми и съответните им двойки съмножители, представени в n-битова разрядна мрежа. Логическата структура може да поеме конвейерното изпълнение на множество последователни операции умножение, от които в конвейера ще се изпълняват паралелно във времето n на брой от тях. Така на всеки такт от конвейера ще слиза поредната двойка съмножители и съответното й произведение. Закъснението на дадено произведение по отношение момента, в който съмножителите се зареждат в конвейера, е равна на n такта. Ако приемем, че едно последователно умножение на 2 числа отнема n на брой такта, то последователното изпълнение на n на брой операции умножение би отнело n^2 на брой такта. Същият брой умножения, с помощта на конвейерно организиран схемен умножител, би отнело $2n$ на брой такта. Този резултат показва, че конвейерната организация, в сравнение с последователната, е в състояние да повиши производителността на процесора при последователно изпълнение на операции умножение, до $n/2$ пъти, т.е. примерно, при

32-битова разрядна мрежа, увеличението достига 16 пъти.

В настоящата работа се представя проектирането и изследването на оригинална логическа структура на конвейерен умножител, чиято организация се отличава от известните по това, че формира междинните суми като паралелни суми от 3 числа, с помощта на концентратор от типа (3:1), чийто синтез и изследване са изложени в [11] и в [12]. Традиционно при умножение (на *n*-битови числа без знак) междинните суми се получават както следва:

$$S_{i+1} = S_i + X.y_{i+1}.2^{i+1} \ , \quad i = \overline{0,n-1} \ . \quad (1)$$

За да натоварим обаче три-входовия суматор, който приемаме да използваме като оператор на дадено ниво в конвейера, трябва да формираме междинната сума както следва:

$$S_{j+1} = S_j + X.y_{i+1}.2^{i+1} + X.y_i.2^i \ . \quad (2)$$

От горния израз се вижда, че във всяка нова междинна сума се натрупват две последователни поразрядни произведения. От тук веднага следва изключително положителният извод, че броят на формираните с помощта на такива суматори междинни суми, ще бъде два пъти по-малък и за техният индекс *j* можем да запишем закона: $j = \overline{1,n/2}$

За логическия синтез на суматора, който ще реализира сумата (2), е полезно тя да бъде илюстрирана със следната схема:

			$q^{(i)}_{n-1}$	$q^{(i)}_{n-2}$	...	...	$q^{(i)}_1$	$q^{(i)}_0$	0	...	0	0	$=X.y_i.2^i$
+		$q^{(i+1)}_{n-1}$	$q^{(i+1)}_{n-2}$		...		$q^{(i+1)}_1$	$q^{(i+1)}_0$	0	0	...	0	0	$=X.y_{i+1}.2^{i+1}$
+	$s^{(j)}_k$	$s^{(j)}_{k-1}$	$s^{(j)}_{k-2}$	...	...	...	...	...	...	...	...	$s^{(j)}_1$	$s^{(j)}_0$	$=S_j$
$s^{(j+1)}_{k+2}$	$s^{(j+1)}_{k+1}$	$s^{(j+1)}_k$	$s^{(j+1)}_{k-1}$	...	...	...	...	...	...	...	$s^{(j+1)}_1$	$s^{(j+1)}_0$	$=S_{j+1}$

Фиг. 1 Схема за натрупване на поредната междинна сума

Схемата показва, че младшите разряди (на брой *2.j*) в *j*-тата междинна сума са окончателно получени при предходните събирания и следователно могат да бъдат пропуснати и изключени от текущото събиране. От тук следва, че всички концентратори употребени в отделните нива на конвейера следва да бъдат с една и съща дължина и съответно изместени наляво един спрямо друг, в съответствие с нарастващия порядък на използваните битовете от множителя (тук се има предвид методът за умножение с младшите разряди напред). Синтезът на принципната логическа схема на тук прилаганите концентратори се различава от този, изложен в [11] и в [12], само по това, че трите числа са разместени едно спрямо друго на един бит, във съответствие с формула (2). Всички останали параметри на концентратора са подробно изложени в посочената литература.

След така изложените съображения следва, че получената структура на конвейерния умножител, реализиращ натрупването (2), ще съдържа два пъти по-малко на брой нива (фиксатори), т.к. на всяко ниво тя консумира едновременно по два бита от множителя. Така, структурата на конвейерен умножител с концентратори от типа (3:1) ще има вида, показан на фигура 2.

Както се вижда от фигура 2, първата междинна сума се формира като сума от следните три поразрядни произведения:

$$S_1 = X.y_0 + X.y_1.2^1 + X.y_2.2^2 \ , \quad (3)$$

а всички останали – според уравнение (2). Вижда се още, че от всяко ниво на конвейера окончателно се получават по две младши цифри на произведението, а сумиращите схеми имат една и съща дължина.

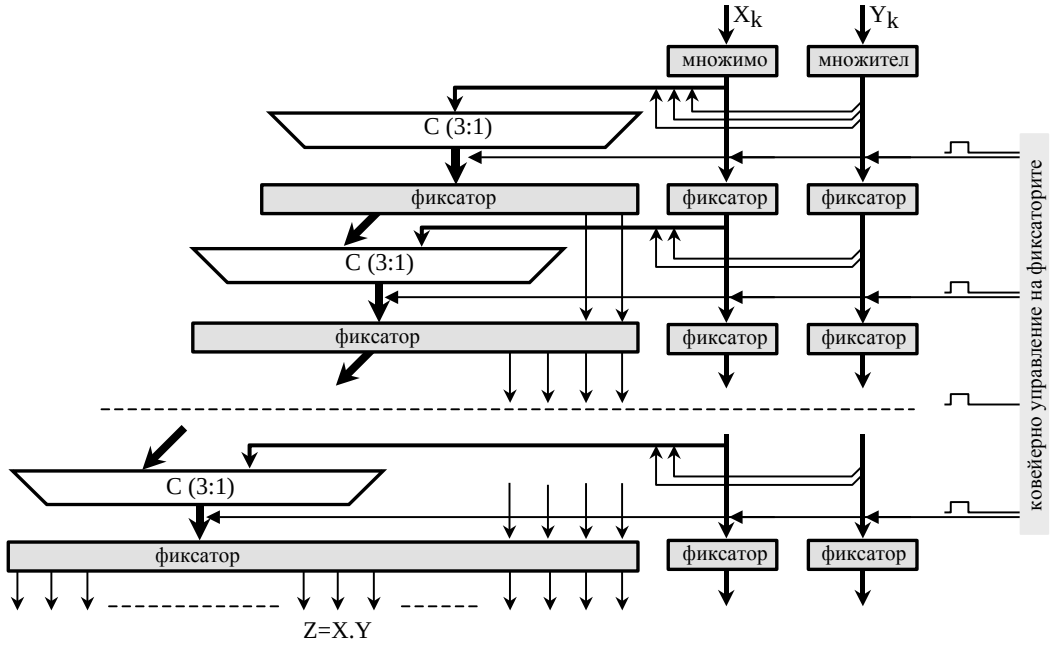
В случаите, когато дължината *n* на разрядната мрежа е четно число, в последната степен на конвейера ще се събират само две числа, от което следва, че в последната степен ще бъде употребен

обикновен двоичен суматор. В случай, че дължината на разрядната мрежа е нечетно число, тогава всички сумиращи схеми ще бъдат еднакви.

Представената конвейерна структура може да се прилага и за умножение на числа със знак, представени в допълнителен код. Единственото изменение, което следва да се направи, се състои в това, че в последната степен следва да се изпълни операция изваждане на множимото, която се налага за корекция на произведението. Това може да бъде реализирано автоматично, чрез управление на входния за суматора мултиплексор с помощта на знаковия бит на множителя y_{n-1} в съответствие с логиката на следния оператор:

$$\begin{aligned} &\textbf{\textit{if}}\ \ y_{n-1}=0\ \ \textbf{\textit{then}}\ \ S_{j+1}=S_j+X.y_{n-1}.2^{n-1}+X.y_{n-2}.2^{n-2} \\ &\textbf{\textit{else}}\ \ S_{j+1}=S_j+\overline{X}.y_{n-1}.2^{n-1}+X.y_{n-2}.2^{n-2}+y_{n-1}\ . \end{aligned} \qquad (4)$$

Такава схема реализира точно алгоритъма за умножение в допълнителен код [10], с тази разлика, че като следствие естествено поставя схемата в класа на схемите за умножение на 2 разряда едновременно.



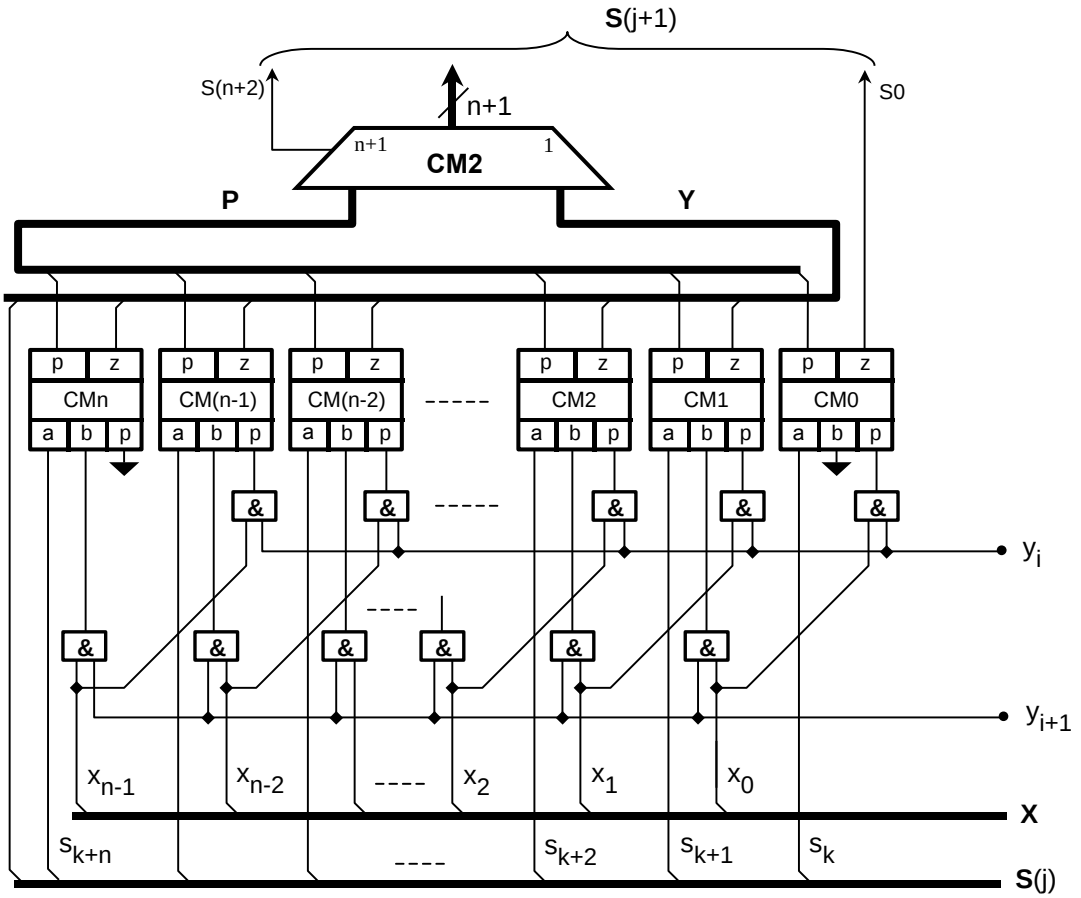
Фиг. 2 Логическа структура на конвейерен умножител

В съответствие с изложените до момента съображения, принципната логическа схема на необходимия за конвейерното приложение концентратор ще има вида, показан на фигура 3.

3. Допълнителни изводи

Всички известни схеми за умножение, за разлика от представената тук, се характеризират с използване на двоичен суматор, в който междинната сума се събира с някакъв еквивалент на поразрядно произведение. Този еквивалент се избира чрез мултиплексиране по единия от входовете на суматора измежду няколко достатъчно сложни за получаване резултата. Броят на тези резултати за всяко ниво в конвейера се движи от 3 до 5 и нагоре (тук се имат предвид схеми, които се получават при използване на алгоритми поне за умножение едновременно с 2 разряда от множителя, с които е естествено да се сравнява тук предложената схема) – [1, 16]. Това означава, че апаратните разходи за реализация на комбинационната част на всяко ниво в конвейерния умножител са най-малко 2 пъти по-вече в сравнение с тези в тук предложената.

По същия начин стои въпросът с оценката на бързодействието на тези схеми. От тук логично следва, че тактовата честота за надеждно и стабилно управление на предложения конвейер с концентартори, ще бъде възможно повишена до 2 пъти.



Фиг. 3 Логическа схема на конвейерния концентратор

4. Заключение

С експериментална цел описаната структура е проектирана конкретно за умножение на 8-битови цели числа, който има 4 нива. Този примерен вариант на проекта е реализиран в технологичната среда WebPack ISE на фирма Xilinx, чрез HDL-езика Verilog, като е предназначен за имплементиране в FPGA-матрица от фамилията Spartan II на същата фирма. На моделната времедиаграма, показана по-долу на фигура 4, може да се види конвейерното умножение на следните последователните двойки съмножители:

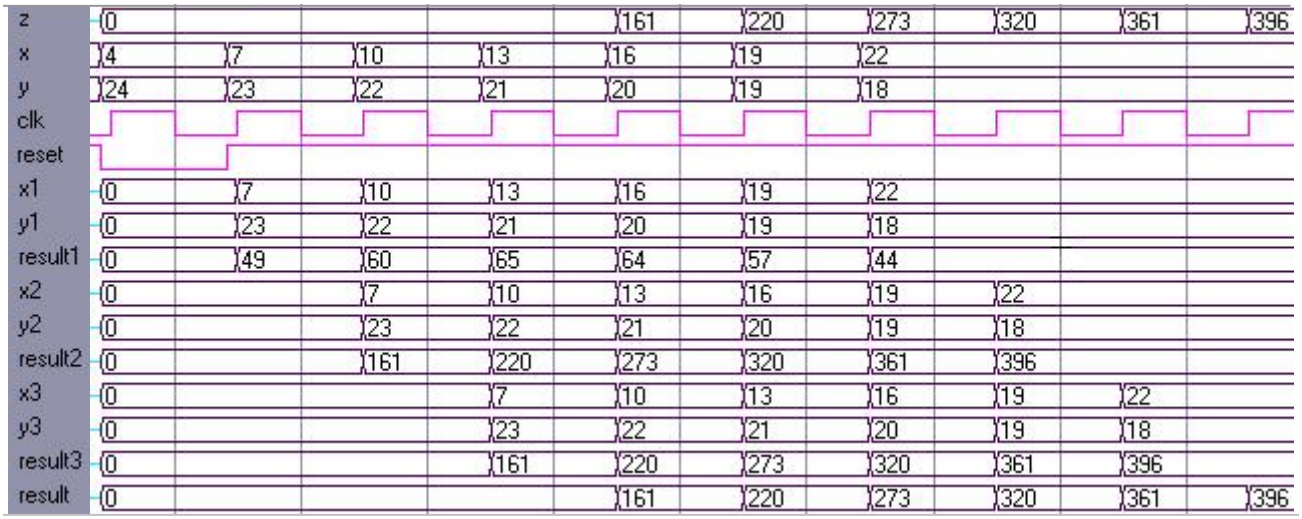
7.23=161 ; 10.22=220 ; 13.21=273 ;

16.20=320 ; 19.19=361 ; 22.18=396 .

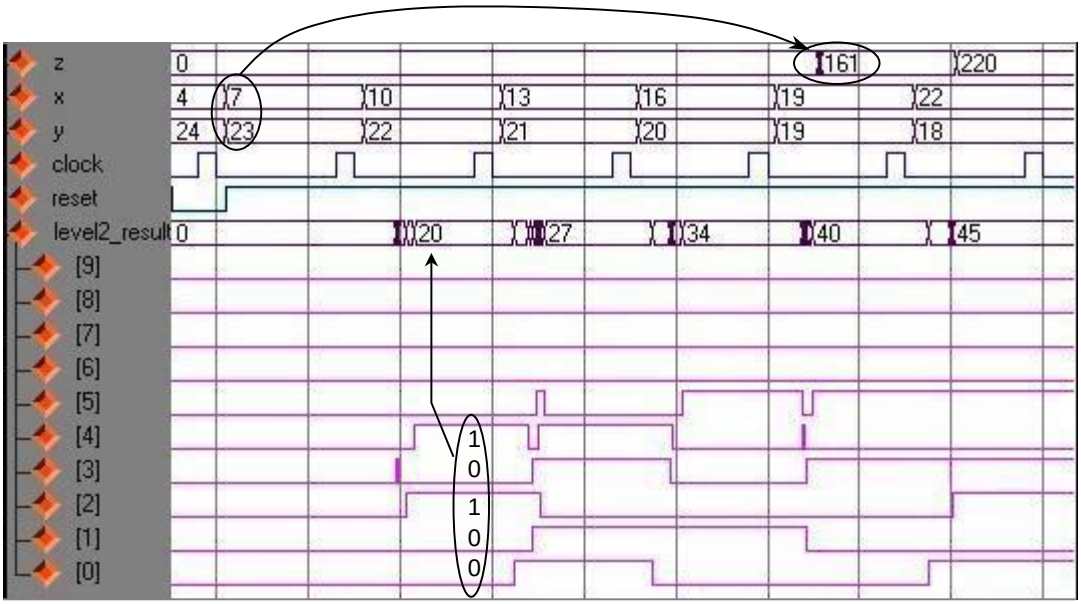
На всеки такт от конвейера слизат посочените произведения. Върху изходите на началните нива на конвейера, т.е. в дълбочина, в десетична бройна система са представени стойностите на междинните суми, до които е достигнало като междинна сума съответното произведение в конвейера.

На следващата фигура 5 е показан преходният процес на изходите на концентратора във второ ниво на конвейера. Така, при движение на двойките съмножители и преминаването им през това ниво, на неговия изход последователно се получават междинните суми 20, 27, 34, 40 и 45.

Тъй като периодът на тактовата последователност в случая е 15[ns] и ясно се вижда, че е възможно да бъде намален двойно, то можем да твърдим, че такъв конвейер ще генерира 16-битово произведение на всеки 7,5[ns]. С други думи повече от 133.10^6 числа в секунда.



Фиг. 4 Изходни и междинни резултати в умножителя



Фиг. 5 Закъснения на второ ниво в конвейера

5. Литература

[1]. Wallace C. S., *Asuggestion for a fast multiplier,* IEEE Trans. On Computers, vol.13, p.14-17, 1964.

[2]. Dadda L., *Some schemes for parallel multipliers,* Alta Frequenza, vol. 34, p.349-356, 1965.

[3]. Waser S., *High-Speed Monolithic Multipliers for Real-Time Digital Signal Processing,* Computer, №10, 1978.

[4]. Baugh Ch., Woley B., *A Two's Complement Parallel Array Multiplication Algorithm,* IEEE Transaction on Computers, C22, №12, 1973.

[5]. Rubinfield L. P., *Proof of the Modified Booth's Algorithm for Multiplication,* IEEE Transaction on Computers, C22, №10, 1975.

[6]. Takagi N., Yasuura H., Yajima S., *High – Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree,* IEEE Transaction on Computers, C34, №9, 1985.

[7]. Карцев М. А., Брик В. А., *Вычислительные системы и синхронная арифметика*, Издательство "Радио и связь", 1981.

[8]. Bickerstaff K., Swartzlander E., Schulte M., *Analysis of column compression multipliers,* 15th IEEE Symposium on Computer Arithmetic, p.33-39, 2001.
[9]. Swartzlander E., Goto G., *Computer arithmetic,* ed. Boca Raton, CRC Press, 2002.
[10]. Тянев Д. С., *Електронни цифрови машини,* ТУ-Варна, ISBN 954-20-0016-2, 1995.
[11]. Тянев Д. С., *Организация на компютъра (цифрова аритметика),* ТУ-Варна, ISBN 954-20-0258-0, 2004.
[12]. Тянев Д. С., Попова С. И., Иванов А. И., Янев Д. В., *Синтез и сравнителен анализ на паралелни многовходови суматори,* сп. "Компютърни науки и технологии" ISSN 1312-3335, №2-2005, стр. 51-61. Интернет публикация: http://www.tyanev.com/resources/docs/Document_V_37_ENG.pdf .
[13]. Kiefer G., Waker A., Thumm A., *Rechnerorganisation,* Institut fur Informatik, Stuttgart, 2001.
[14]. Schiller, Jochen, *Rechnerstrukturen,* Freie Universitat, Berlin, 2003.
[15]. *Design FPGA-Based DSPs for Performance and Power*, Jim Simkins and Ben White, "Chip Design Magazine", February, 2005.
[16]. *Multiplication in FPGAs* - www.fpga-guru.com/multipli.htm

За контакти:
доц. д-р инж. Димитър С. Тянев
катедра "Компютърни науки и технологии"
Технически университет – Варна
WEB: www.tyanev.com

РАЗПРЕДЕЛЕНА СИСТЕМА ЗА УПРАВЛЕНИЕ НА СТЪПКОВ ДВИГАТЕЛ

Димитър С. Тянев, Александър И. Иванов, Драгомир В. Янев, Стефка И. Попова

Резюме: Представено е едно ново апаратно-програмно решение на система за управление на стъпков двигател с разпределени функции и ресурси, който е изпълнителен орган в разрязващия агрегат на ролетно печатаща машина. Устройството представлява ниво в разпределена компютърна система за цялостно on-line управление на печатащата машина. Проектирането и реализацията на устройството е изпълнено със средства и интегрални елементи на фирма Xilinx. Управлението замества морално остаряла електромеханична система за разрязване, като постига изключително висока точност.

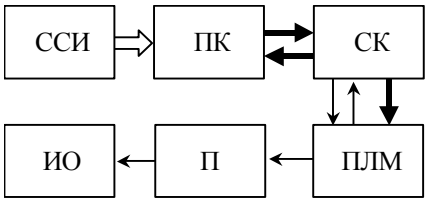
Distributed System for Stepper Motor Control

Dimitar S. Tyanev, Aleksandar I. Ivanov, Dragomir V. Yanev, Stefka I. Popova

Abstract: In this document is presented a novel aparatus-programing solution for controling device for a stepper motor, which is the actuator in the splitting unit of a roll printing machine. The device is a part of a distributed computer system for online control of the printing machine. The design and implementation of the device is done using the tools and programmable devices of Xilinx. This type of control replaces the outdated electro-mechanical slitting control system, thus achieving higher level of precision.

1. Увод

В настоящата работа се предлага едно ново апаратно-програмно решение на система за управление на стъпков двигател с разпределени функции и ресурси, характеризираща се с това, че реализира управляващия алгоритъм хардуерно, на базата на програмируема логическа матрица. Общата структура, в състава на която се намира подсистемата за управление на стъпковия двигател, е показана на фигура 1. Тя съдържа система за сбор на инфор-мация (ССИ); персонален компютър (ПК); специализиран контролер (СК); програми-руема логическа матрица (ПЛМ), в която е реализирано устройството за управление на стъпковия двигател (УУСД); преобразувател (П) и изпълнителен орган (ИО). Изпълнителният орган, който се има предвид, е двуфазен реверсивен стъпков двигател тип PK264A2-SG7.2, с предавателно отношение 7,2:1, който се задвижва чрез преобразувател тип CSD2120-T.



Фиг. 1 Общ вид на подсистемата

Общата схема на алгоритъма, по който функционира представената на фигура 1 система, е показана на фигура 2.

Фиг. 2 Общ вид на софтуера на подсистемата

Както ПК така и МК представляват многофункционални апаратно-програмни системи, в които задачата за управление на стъпковия двигател се решава при необходимост в режим на прекъсване.

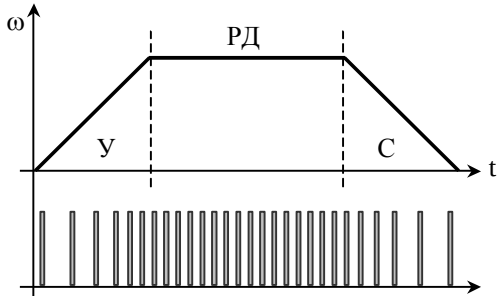
2. Синтез на управляващия алгоритъм

Тъй като бързодействието за отработване на отклонението в работния орган на системата е желано качество, за управлението на двигателя е избран линеен закон за скоростта ω, който съдържа етап на ускорение (У), етап на равномерно движение (РД) и етап на спиране (С):

Ъгловата скорост ω и честотата f на управляващите импулси са линейно зависими:

$$\omega = \alpha.f \ , \qquad (1)$$

където с α е означен ъгълът на завъртане на ротора на двигателя. Тъй като ъгловата стъпка на завъртане s е константа (s=0,25°), следва че при управление манипулираната величина е честотата. Реалният сигнал за управление представлява импулсната последователност, показана на фигура 3.



Фиг. 3 Закон на скоростта

Продължителността на правоъгълните импулси е постоянна и е ограничена от преобразувателя до 5[μs]. При смяна на посоката на движение на двигателя, управлението следва да осигури след последния и преди първия управляващи импулси, празен интервал с минимална продължителност от 100[μs]. Максималната скорост на въртене на двигателя е 250[Rpm]. За постигане на тази скорост импулсите трябва да следват с минимален период, който определя максималната честота, както следва:

$$f_{max} = \frac{360^\circ}{s}.250.\frac{1}{60} = 6000[\text{Hz}] \ , \ (2)$$

Така за периода се получава:

$$T_{min} = \frac{1}{6000} = 167\ [\mu\text{S}] \ . \qquad (3)$$

За да се постигне линейност при нарастване на скоростта, генерирането на управляващите импулси трябва да се извършва променливо закъснение. Всяко следващо закъснение трябва да е такова, че постигната скорост на въртене на двигателя, да следва профила на ускорението възможно най-точно, както е показано на фигура 3. При достигане на максималната скорост честотата на управляващите импулси остава постоянна, а при спиране те следват един след друг с такива закъснения, с каквито съответно са се подавали по време на ускорителния етап, но в обратен ред.

2. Изчисляване на данните за управление

Времевото закъснение δt, с което се появява всеки импулс, управлява скоростта, т.е. колкото по-малко е то, толкова по-голяма е скоростта. Движението на стъпковия двигател е образувано от дискретни стъпки, а зърнеността на това движение се определя от честотата на таймер. Параметрите на таймера и механичното движение са свързани чрез следната зависимост:

$$\delta t = c.t_t = \frac{c}{f_t} \ , \qquad (4)$$

където t_t и f_t са съответно периодът и честотата на таймера, а с c е означен междустъпковият интервал в брой негови периоди. За да генерираме реално поредица от импулси е необходима базова честота, чрез която да изразим дължината на интервалите между стъпките. Този таймер е реализиран в програмируемата логическа матрица ПЛМ, поради което изчислението на елементите на масива от периоди за управляващите импулси, следва да се извърши спрямо честотата, на която работи таймерът – f_t.

За да се осъществи управлението на двигателя по зададения закон, е необходимо да се изчислят закъсненията на последователните управляващи импулси, които като елементи на един едномерен масив ще представляват данновата основа, върху която устройството в ПЛМ ще генерира тези импулси. Преместването *S*, което извършва двигателят за времето t_n, за което достига максималната скорост, се определя според закона:

$$S = 0{,}5.\dot{\omega}.t_n^2 , \qquad (5)$$

където с $\dot{\omega}$ е означено допустимото за системата на двигателя максимално ускорение.

Изминатият път се изразява още така:

$$S = \alpha.n , \qquad (6)$$

т.е. това е ъгълът на завъртане при една стъпка умножен с броя на стъпките. Приравнявайки (5) и (6) определяме броя на стъпките за времето на ускорение t_n:

$$n = \frac{\dot{\omega}.t_n^2}{2.\alpha}. \qquad (7)$$

След като е известен броят на стъпките за достигане на максималната скорост, следва да се определят стойностите на последователните закъснения. Можем да изразим времето за извършване на *k* на брой стъпки по горната формула така:

$$t_k = \sqrt{\frac{2.k.\alpha}{\dot{\omega}}} . \qquad (8)$$

Интервалът между две последователни стъпки се изразява чрез формулата:

$$T_k = t_{k+1} - t_k = \sqrt{\frac{(k+1).2.\alpha}{\dot{\omega}}} - \sqrt{\frac{k.2.\alpha}{\dot{\omega}}} . \qquad (9)$$

За да намерим времето за изпълнение на първата стъпка, което се явява първият период T_0 на импулсната поредица, заместваме *k* с нула:

$$T_0 = \sqrt{\frac{2.\alpha}{\dot{\omega}}} - 0 = \sqrt{\frac{2.\alpha}{\dot{\omega}}} .$$

За определяне на всеки следващ период T_k, използваме общата формула, в която след полагане на частта, съответстваща на определеното по-горе за T_0, получаваме следния вид:

$$T_k = T_0.(\sqrt{k+1} - \sqrt{k}) . \qquad (10)$$

Имайки предвид, че T_k е свързано, според (4), с междустъпковия интервал *c*, получаваме формулите за изчисление на тези интервали, съответно c_0 и c_k:

$$c_0 = \frac{1}{t_t}\sqrt{\frac{2.\alpha}{\dot{\omega}}} \quad \text{и} \quad c_k = c_0.(\sqrt{k+1} - \sqrt{k}). \qquad (11)$$

Следва да изтъкнем, че между стъпковите интервали следва да се изчисляват с точност до единица, тъй като имат характера на цели числа – те представляват дължината на интервалите в брой цикли на таймера. Изчислението на поредицата коефициенти c_k според получената формула е много бавна процедура, тъй като това се налага да се извърши програмно в архитектурно слабата операционна част на микроконтролера. Ето защо за това изчисление се търси друга по-подходяща форма. Чрез апроксимация на елементарната функция корен втори в (11) с ред на Тейлор, стойността на коефициентите може да бъде получена чрез следната рекурентна формула [1, 2]:

$$c_k = c_{k-1} - \frac{2.c_{k-1}}{4.k+1} . \qquad (12)$$

Това изчисление е многократно по-бързо от изчислението (11) с двата квадратни корена, но въвежда грешка 0,44 при *k*=1, която компенсираме като умножаваме c_0 с константата 0,676.

Разполагайки с броя *n* на стъпките за ускорение, с изчислен първи елемент c_0 и с формулата за всеки следващ елемент c_k, профилът на функцията за управление от фигура 3 е напълно определен.

3. Програмен модел на УУСД

За да поясним кратко алгоритъма за управление и структурата на устройството за управление на стъпковия двигател ще представим програмния модел, който "вижда" микроконтролерът. Последният е свързан с УУСД с 10-битова адресна и 8-битова даннова шини. Така в адресното пространство от 1[KB] програмният модел заема част от следните области, показани на фигура 4:

Фиг. 4 Програмен модел на УУСД

Във втората половина на адресното пространство е разположена RAM-паметта на УУСД, в която 512 клетки са предназначени за съхранение на еднобайтовите елементи на масива от стъпковите интервали c_k, изчислени от микроконтролера СК, според формулите (11) и (12). Масивът със стъпковите интервали обаче се записва в тази част на паметта в минимизиран обем по следния начин – реално записаните елементи представляват стъпковите интервали само за етапа У на закона за управление (вижте фигура 3). Това е възможно, защото в етапа С броят на стъпковите интервали е същият, както в етапа У, а стъпковите интервали в етапа РД съвпадат със стойността на последния стъпков интервал от етапа У.

В първата половина на адресното пространство са разположени операционните регистри на логическата структура на УУСД. Започвайки от адрес 000h, техния ред е следния: регистър на командата (Reg.I) с дължина 1[B]; регистър за брой на стъпките при ускорение Reg.N с дължина 2[B]; регистър за брой на стъпките при равномерно движение Reg.M с дължина 4[B]; регистър за фактор на деление Reg.FD с дължина 1[B]; регистър за ширина на импулса Reg.W с дължина 1[B]; контролен регистър Reg.C с дължина 4[B]. Регистрите Reg.FD и Reg.C не са показани в логическата структура, тъй като тяхното предназначение е да осигурят универсалност на устройството за управление, по отношение на техническите параметри на различни от споменатия тип стъпкови двигатели.

Логическата структура на реализираното УУСД е представена на фигура 5.

4. Алгоритъм за управление

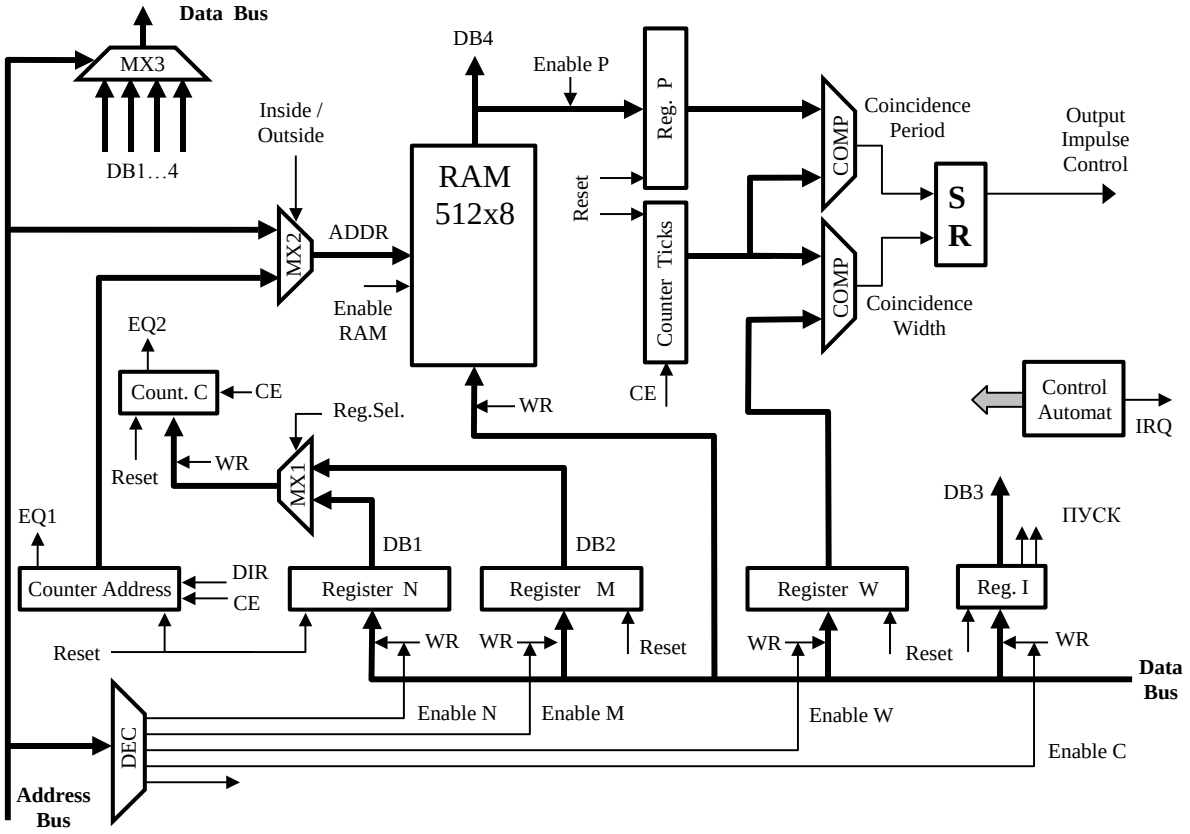
При включване на захранването системата се установява в изходно състояние. В това състояние адресната шина на RAM-паметта е включена чрез MX2 към адресната шина на микроконтролера. Така регистрите и паметта в УУСД са видими за микроконтролера, който го инициализира със следните константи:

- Фактор на деление. Това е цяло 8-битово число, което характеризира времевите параметри на използвания тип стъпков двигател. Съдържа се в Reg.FD ;
- Ширина на управляващия импулс, според изискванията на използвания тип стъпков двигател. Представлява цяло 8-битово число в регистър Reg.W.

Посочените константи се записват в УУСД еднократно от микроконтролера.

Нов сеанс за управление на стъпковия двигател започва по инициатива на по-високото ниво в разпределената система, т.е. със запис на всички необходими нови данни в регистрите на УУСД и в RAM-паметта. Подготвителната процедура изпълнява записите в следната последователност:

1. Запис в регистър Reg.N на броя стъпки в етапа за ускорение (цяло 2-байтово число) ;
2. Запис в регистър Reg.M на броя стъпки в етапа за равномерно движение (цяло 4-байтово число) ;
3. Запис в RAM-паметта на едномерния масив от еднобайтови цели числа, съответстващи на междустъпкови интервали



Фиг. 5 Логическа структура на УУСД

c_k , изчислени от микроконтролера ;

4. Запис в регистъра на командата Reg.I на флага за посока на движението (бит №1) ;
5. Реализация на пауза при смяна на посоката на движение на стъпковия двигател, която е с продължителност не по-малка от 100[μs] за посочения тип стъпков двигател (СД).
6. Запис в регистъра на командата Reg.I на флага “ПУСК” (бит №0) .

С последния запис на данни в УУСД от подготвителната процедура вътрешния управляващ автомат СА стартира поредния сеанс за управление на стъпковия двигател.

Флагът “ПУСК”, който се появява в регистъра на командата Reg.I, разрешава изхода “Output Impulse Control” и извежда управляващия автомат СА от изходно състояние. Алгоритъмът за управление минава през следните етапи: етап У, етап РД, етап С (вижте фигура 3). Във всеки един от тези етапи се генерират стъпкови интервали с по един управляващ импулс. Генерирането на стъпковия интервал (период) започва с генерирането на единичния управляващ импулс. Генерирането на стъпковия интервал представлява генериране на продължителността на правата (единичната) фаза, след което следва генериране на продължителността на инверсната фаза. Така двете фази формират цялостно управляващия такт. Продължителността на времевите интервали се измерва с периода на основния тактов генератор. Импулсите на този тактов генератор модифицират съдържанието на всички броячи, които измерват времеви интервали.

Генерирането на правата фаза се осъществява от изходния RS-тригер. Управлението на този тригер се осъществява от сигналите “Coincidence Period” (CP) и “Coincidence Width”

(CW). Тези сигнали се получават в резултат на сравнение на съдържанието на таймера Counter Ticks, измерващ времето в текущия период, с двете гранични стойности: продължителност на управляващия сигнал, намираща се в Reg.W и продължителността на текущия период, намираща се в Reg.P. За всеки период е в сила отношението (Reg.W) < (Reg.P). Тъй като изходното състояние на RS-тригера в началото на всеки период е единица, тази единица се поддържа до момента, в който се появи сигналът "CW", т.е. до постигане на равенството (Counter Ticks) = (Reg. W). С появата на сигнала "CW" RS-тригерът се превключва в нула. Този сигнал е с продължителност равна на един период на времевия тактов генератор. Така започва генерирането на инверсната фаза в текущия период. Тя приключва когато съдържанието на брояча Counter Ticks се изравни със съдържанието на Reg.P. В този момент, когато (Counter Ticks) = (Reg.P) възниква сигналът "CP". Този сигнал бележи края на текущия период и същевременно началото на следващия период. Продължителността на този сигнал също е равна на един период на времевия тактов генератор. Сигналът "CP" се използва и за нулиране на брояча Counter Ticks.

Генерирането на двете фази на следващия период следва да започне при ново съдържание на регистъра на продължителността на периода Reg.P. Новото съдържание е подготвено (прочетено) от RAM-паметта предварително, по време на вече завършилия период. Това е постигнато чрез модификация на адресния брояч Counter Address. Модификацията на този брояч е в положителна посока по време на етапа У, липсва по време на етапа РД, и е в отрицателна посока по време на етапа С от закона за управление. Записването на продължителността на новия период в Reg.P се извършва автоматично от управляващия автомат СА, с което започва новото генериране. Ако генерираният период е последен, управляващият автомат генерира сигнала за край (сигнал за прекъсване) "IRQ".

Моментите за модификация на режимите на адресния брояч се съобщават на управляващия автомат чрез измерване на продължителността на отделните етапи за управление на стъпковия двигател – У, РД и С. Самото измерването става с помощта на брояча на цикли Counter C и дешифратора на неговото нулево съдържание, който формира сигнала "EQ2". Всяко начално съдържание на този брояч се зарежда през мултиплексор MX1 от регистър Reg.N или от регистър Reg.M. Този мултиплексор се управлява от сигнала "Register Select".

След генерирането и на последния управляващ импулс, състоянието на структурата на УУСД е следното: адресният брояч Counter Address се намира в изходно състояние (сочи клетката, в която се намира първият елемент от масива); броячът на цикли Counter C е нулиран; таймерът Counter Ticks е нулиран и RS-тригерът е в единично състояние; Управляващият автомат СА е завършил алгоритъма за управление и се намира в изходно състояние, при което MX2 е превключен към входната адресна шина, регистърът на командата Reg.I е нулиран; Reg.N, Reg.M и Reg.P запазват старото си съдържание.

Логическата структура на УУСД е проектирана така, че чрез мултиплексора MX3 могат да бъдат прочитани с цел контрол съдържанията на част от регистрите от структурата на УУСД.

4. Заключение

Описаното устройство за управление на стъпков двигател е проектирано в технологичната среда WebPack ISE на фирма Xilinx, чрез HDL-езика Verilog. Реализацията му е в FPGA-матрица от фамилията Spartan II на същата фирма. Устройството за управление на стъпковия двигател е част от структурата на

специализирана компютърна система, която е самостоятелна част от разпределената система, показана на фигура 1. Тази част от разпределената система е предназначена за управление на разрязващия агрегат на ролетни печатащи машини. Реализираното on-line управление на разрязващия агрегат, в сравнение с традиционно използваната елекромеханична система, позволява да се постигне изключително висока точност, оценявана с отклонения в интервала ±0.2[мм].

5. Литература

[1]. *AVR – Linear speed control of stepper motor,* Atmel 8-bit Microcontrollers Application Note, Rev. 8017A-AVR-06/2006.

[2]. Austn, D., *Generate stepper-motor speed profiles in real time*, Embeded System Programming, January 2005.

[3]. Тянев Д. С., *Организация на компютъра – проектиране на логически структури,* ТУ-Варна, ISBN 954-20-0259-9, 2004.

[4]. ISE 8.1i – *Libraries Guide*, Xilinx.

[5]. Bansker J., *Verilog HDL Synthesis a Practical Primer*, Star Galaxy, 1998.

За контакти:
доц. д-р инж. Димитър С. Тянев
катедра Компютърни науки и технологии
Технически университет – Варна
WEB: www.tyanev.com

ГЕНЕТИЧНИТЕ АЛГОРИТМИ ПРИ УПРАВЛЕНИЕ НА ПРОЕКТИ

Милена Н. Карова, Виолета Т. Божикова

Резюме: В доклада се предлага нов метод за управление на проекти, в частност съставяне на разписание на проект и управление на ресурсите на проект при зададени ограничения. Ограниченията са свързани с определени срокове и бюджет, ефективно разпределение на ресурси и отчитане на технологичната зависимост между отделните компоненти на проекта.
Тези основни проблеми са отчетени при планиране на проекта чрез използване на Генетичен Алгоритъм.

A Genetic Algorithm for Multi-Project Problem

Milena N. Karova, V. T. Bozhikova

Abstract: This paper presents a Genetic Algorithm for the Multi-Project Problem. There are strict requirements to timetable and budget cost for projects. The objective is to minimize the makespan of the project. It proposes a new method with chromosome representation, based on random keys. The schedules are constructing using a heuristic that builds parameterized active schedules based on priorities, delay times, release dates, defined by the genetic algorithm.

1. Увод

Проект е мероприятие което включва в себе си набор от действия, изпълнението на които е необходимо за да се изпълнят целите на проекта.

Управлението на проекти е процес със сложно решение, включващо непрестанно противопоставяне между стойност и време. По принцип проблемът е съчетание от решения в областта на планирането и разписанието.

Планирането основно е стратегически процес с изискване за основни типове ресурси на всеки времеви период от планирането. Създават се ресурсни профили, съобразени с изисквания и ограничения по отношение на :

- Какви задачи трябва да бъдат изпълнявани и в какъв ред за да се получи крайния резултат от проекта.
- Кой ще изпълни тези задачи.
- Колко ще струва това.

Съставянето на разписание включва установяване местоположението на дадените ресурси, определяне началното време и времената за отделните дейности. Определя се нивото на оптималност или се определя кое решение е добро.

2. Постановка на задачата

Проектът може да бъде разглеждан като съвкупност от ресурси и задачи. Ресурсът се характеризира с нормално количество потребления на ден (в единица ресурс), брой потребления на ден с отчитане на свръхпотребление, стойност на единица ресурс при нормално и свръх-потребление.

Дейността се характеризира с продължителност, начална и крайна дата, списък от разпределени ресурси, списък от предишни и последващи дейности.

Дейностите могат да се разделят на 2 типа, в зависимост от разчета на тяхната продължителност:

- Продължителност на работата, определена по задание.
- Брой единици ресурс необходими на ден (ако те са разпределени трябва да са достъпни едновременно).

- Продължителност на работата, определена от ресурсите. Необходимостта от ресурси се изразява в определени количества ресурси, необходими за изпълнение на поставените задачи.

Съставяне на разписание на дейностите по проекта може да се извършва с помощта на ГА, чиято задача е да намери добро (оптимално) решение.

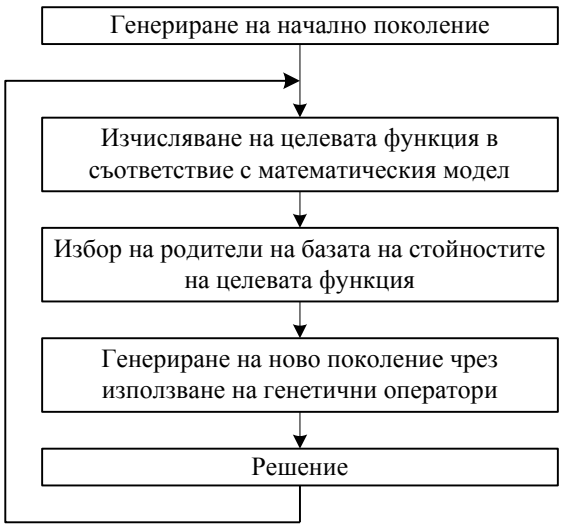
3. Основни положения при разработка на ГА

При разработката на ГА се минава през следните етапи:

- Определяне кодировката на хромозомите в популацията (двоична, директна, как се кодират гените).
- Определяне на целевата функция, чрез която ще се оценява пригодността на всяко решение. Необходимо е тя да отчита всички поставени ограничения.
- Определяне на началната популацията от хромозоми (потенциални решения).
- Определяне правилата по които ще се формира всяко следващо поколение (генерация).
- Определяне типа на генетичния оператор кросовър: едноточков, многоточков, аритметичен, цикличен, униформен и др. Определяне на коефициента на кросовър.
- Определяне типа на генетичния оператор мутация: едноточкова, вероятностна и др. Определяне коефициента на мутация.
- Определяне типа на генетичния оператор селекция: избор на хромозомите с най-добра стойност на целевата стойност, използване на елитизъм и др.

На Фиг. 1 е показана обобщена схема на Генетичен Алгоритъм (ГА).

Всеки ген трябва да е уникален, ако се предлага кодировка, при която алелът на гена представлява номер на дейността (операцията), която трябва да се извърши, а локусът – времевия слот на извършване на операцията.



Фиг. 1

Уникалността произтича от факта, че всяка операция трябва да се включи задължително в разписанието и трябва да присъства само веднъж.

Изискването за уникалност най-често се нарушава при работа на оператора кросовър.

Ограниченията се свеждат основно до ресурсни и времеви ограничения. По тази причина трябва да се формулират ясни правила и броят на ограниченията да се сведе до минимум. Примерно:

- Избор на най-кратката дейност.
- Избор на най-дългата дейност.
- Избор на най-евтината дейност.
- Забрана на дейност със свръхпотребление на ресурси.
- Избор на дейност със свръхпотребление на ресурси.

Освен за уникалността на алелите на гените, ГА трябва да съблюдава последователността на операциите (отношението „предшественик-последовател"). В повечето случаи операциите са свързани в някаква последователност и тя не бива да се нарушава.

Възможно е премахването на ограничението „предшественик последовател", ако се промени кодировката на гени-те. Алелите да бъдат времеви слотове, а локусите – номерата на операциите.

В този случай може да отпадне и изискването за уникалност на гените, защо-

то е възможно една операция да се извърши в няколко времеви слота.

Всяко правило може да се окаже най-изгодно. Най-кратката дейност трябва да се избира тогава, когато работата претендент за включване в разписанието има времетраене, определено от ресурси. При недостатък от ресурси, такава дейност се включва за да не се увеличава продължителността на проекта.

По принцип най-дългите дейности се поставят в началото на проекта (тогава натоварването на ресурсите не е толкова голямо). Най- евтината дейност е изгодно да се избере, когато приоритет се явява стойността на проекта. Забрана или разрешение на свръхресурси може да повлияе и на продължителността на проекта и на стойността му. От една страна свръхресурсите позволяват да се съкрати продължителността на отделните дейности и на проекта като цяло, от друга страна – нарастват разходите (наднорменото заплащане по правило е над стандартните разценки).

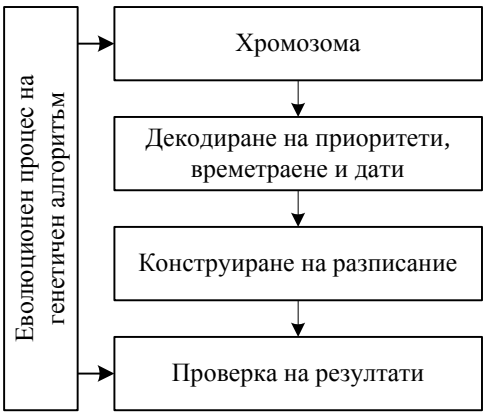
4. Архитектура на ГА.

ГА генерира хромозомите на всяка следваща популация на 2 фази: кодира реда на изпълнение, времетраене и начална дата и втората фаза – определя разписанието на дейностите (Фиг. 2).

Така хромозомата може да се представи с 2*n+m, където n е броя на дейностите, m – броя на проектите.

Chromosome=($gene_1$,…,$gene_n$, $gene_{n+1}$,…,$gene_{2n}$, $gene_{2n+1}$,…,$gene_{2n+m}$)

Първите n гена определят приоритетите на всяка дейност. Средните n+1, 2n се използват за определяне продължителността на всяка дейност. Последните m гена определят началната дата на всеки проект (дейност). Първите n гена имат стойности между 0 и 1. Приоритетът се определя по различен начин в зависимост от ресурсите, важ-ността и стойността.



Фиг. 2

Вторите гени имат стойности между 1,0 и 2,0. Те се определят в зависимост от възможната максимална продължителност на дейността и усреднен експериментален времеви коефициент.

Последните гени се изписват като четирицифрено число, представляващо ден и месец на възможното стартиране на проекта (дейността).

Размерът на популацията може да се определи по различен начин. Предлага се минималния размер да бъде 0,2*броя на дейностите в проекта (0,2*n).

Възможни са различни приложения на генетичните оператори: селекция, кросовър и мутация. Селекцията и кросоврът определят кои родители (хромозоми) ще участват при създаването на следващото поколение.

При избора на участници в следващото поколение може да се приложи стратегията на елитизма, при който 10% от най-добрите хромозоми се копират като елементи на следващото поколение. Така за добрите решения не съществува опасност да се загубят. Качеството на популацията се подобрява, възможността за по-бързо достигане на добро решение (по-бърза сходимост) се увеличава.

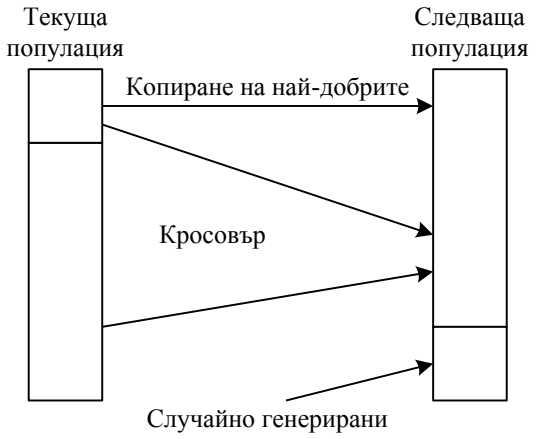
Вместо традиционните едноточков и двуточков кросовър, се предлага използване на униформен кросовър, който предлага много добри резултати [дисертация цитиране]. От всички хромозоми на популацията (включително и тези, които са

копирани по време на елитизма) случайно се избират двама родители. Използва се маска от 0 и 1, като стойността 1 определя „детето” да вземе гена от първия родител, а стойността 0 – от втория. На Фиг. 3 е дадена схема със пример з използване на униформен кросовър.

Маска	1	1	0	1	0
Родител1	0,57	0,93	0,36	0,12	0,78
Родител2	0,46	0,35	0,59	0,89	0,23
Поколение1	0,57	0,93	0,59	0,12	0,23

Фиг. 3

Кросовърът се извършва с вероятност 70%. Мутацията използва нисък коефициент на мутиране 20%. Предлага се едноточкова мутация, където алела на случаен ген се замества с валиден нов алел или два гена си разменят алелите. Ролята на мутацията се състои в пре-дотвратяване на преждевременна сходимост (ако всички хромозоми станат еднакви).Процесът на създаване на всяка следваща генерация е даден на Фиг. 4.



Фиг. 4

Целевата функция се изчислява като сума от най-дългото времетраене (T_i), най-късото времетраене (E_i) и текущото отклонение във времето за проекта (FD_i).

$$F = \sum_i T_i + \sum_i E_i + \sum_i FD_i \qquad (1)$$

Като критерий за преустановяване работата на ГА се предлага достигането на добро (оптимално) решение или 50 генерации.

5. Заключение

Модификации на разглежданата задача могат да се получат чрез добавяне на нови ограничения и нови понятия. Възможно е промяна на целевата функция с добавяне на стойност на проекта. По отношение на ГА е възможно да се увеличи броя на предлаганите генетични оператори, промяна на генетичните коефициенти и промяна на критерия за преустановяване работата на ГА.

Подобни на предлаганите методи за управление на проекти ще се развиват, тъй като в организациите с проектно-ориентирана дейност съществува необходимост от програмни средства за управление на проекти. Те биха обезпечили намирането на оптимален начин за реализиране на проекта по време и стойност при максимално ефективно използване на ресурси.

Литература

1. Jalote P., Software Project Management in Practice, Addison Wesley, 2002
2. Stover T., Microsoft Office Project 2003 Inside Out, Microsoft Press, 2004
3. Stover T., Lowery G., Managing Projects with Microsoft project 2000
4. Карова М. Изследване и реализация на генетични алгоритми за решаване на един клас задачи, докторска дисертация, Варна, 2006.
5. Уткин Е.А., Кравченкор В.П., Проект-менеджмент, Издателство „ТЕИС”, Москва, 2002
6. MS Project 2003, Step by Step, Софтпрес, 2005
7. http://www.pmi.org.
8. http://www.project.bg

За контакти:
гл. ас. д-р инж. Милена Н. Карова
гл. ас. д-р инж. Виолета Т. Божикова
катедра Компютърни науки и технологии
Технически университет – Варна
E-mail:mkarova@ieee.bg
E-mail:vbojikova2000@yahoo.com

ОТКРИВАНЕ И ОБРАБОТКА НА ПРЕХОДИТЕ ПРИ МАЩАБИРАНЕ НА ЗВУКОВИ СИГНАЛИ ВЪВ ВРЕМЕТО ЧРЕЗ ФАЗОВ ВОКОДЕР С ТВЪРДО БЛОКИРАНА ФАЗА

Лъчезар Ил. Георгиев

Резюме: В доклада се предлага метод за откриване и обработка на преходите при мащабиране на звукови сигнали във времето чрез фазов вокодер с твърдо блокирана фаза. Приема се, че има преход:

а) ако спектралната енергия нараства в два последователни кадъра и едновременно с това

б) енергийното съотношение на текущия (последен) към предпредпоследния кадър надхвърля праг *R*.

Така откритият преход се обработва, като в кадъра, в който той е открит:

а) изходната фаза на каналите-върхове се „замразява“, т. е. се прави равна на входната,

б) откриването на преходи за следващите *n* кадъра се забранява,

б) амплитудата на изходния сигнал се увеличава $\sqrt{2}$ пъти.

Transient Detection and Processing in Audio Signal Time-Scaling by Rigid-Phase-Locked Vocoder

Lâtchezar I. Georgiev

Abstract: The paper offers a method for detection and processing of transients in time-scaling of audio signals via rigid-phase-locked vocoder. A transient is assumed:

a) if the spectral energy increases in two consecutive frames and at the same time

b) the energy ratio of the current (last) and last-but-two frames exceeds a threshold *R*.

The so detected transient is processed by executing the following steps in the frame in which it is detected:

a) the output phase of the peak channels is „frozen“, i. e. it is made equal to the input phase,

b) transient detection for the following *n* frames is disabled,

c) the amplitude of the output signal is increased $\sqrt{2}$ times.

1. Увод

При цифровата обработка на звукови сигнали, една от основните операции е мащабирането във времето без промяна на височината на тона. Един от най-добрите и не особено сложни от изчислителна гледна точка методи за независима промяна на мащаба във времето и височината на тона е *фазовият вокодер* [1]. За съжаление при него съществуват два характерни проблема:

1) фазови изкривявания (т. нар. „фазовост“)

2) изглаждане („размазване“) на преходите.

В [2] е разгледана перспективна негова разновидност – т. нар. фазов вокодер с „твърдо блокиране на фазата“, при която се отстранява почти напълно първият проблем и донякъде вторият. Но съществуващите методи за по-добро решаване на втория проблем или водят до грешки в мащаба на времето [7], или са твърде сложни [3–6] и приложението им във фазовия вокодер би обезсмислило едно от предимствата му – относителната простота, или техният алгоритъм не е изяснен от автора [8, 9],

което, разбира се, не изключва неговата сложност!

В настоящия доклад се предлага прост метод за откриване и обработка на преходите, който не страда от посочените по-горе недостатъци и позволява достатъчно сигурно и вярно откриване на преходите и обработка, при която значително се подобрява слуховото възприятие. Той е специално предназначен за използване във фазов вокодер с твърдо блокиране на фазата и е много подходящ за работа в реално време.

2. Откриване на преходите

Предложеният тук метод за откриване на преходите е от групата на енергийните методи и се нарича „метод на достатъчния устойчив растеж“. Съгласно този метод се приема, че е открит преход, ако са изпълнени едновременно следните две условия:

1) Спектралната енергия нараства в два последователни кадъра

2) Енергийното съотношение на текущия (последен) към предпредпоследния кадър е > R

където R е определен по опитен път праг, който се наглася в обхвата от 1S до 2. За минимизация на лъжливите откривания на преходи най-подходящата стойност е 2, а за минимизация на пропуснатите преходи – 1S. (По подразбиране, $R = 2$.)

Заб. Първоначално, второто условие беше разликата, а не съотношението на енергиите да е по-голяма от даден праг, но се установи, че така вероятността за пропускане на преход се увеличава при тихи сигнали, а за лъжливо откриване – при силни, което прави невъзможно определянето на универсален праг. Затова се взе решение разликата да се замени със *съотношение*, при което се избягва тази зависимост от динамиката и вече има възможност да се избере праг, при който алгоритъмът да работи с минимален брой както на пропуснатите преходи, така и на лъжливите откривания на преходи.

3. Обработка на така откритите преходи

Предложеният тук метод за обработка на преходите се нарича „замразяване на фазата“. За разлика от „замразяването“ на кадъра [7], при него се запазва само фазата, и то само на спектралните върхове.

Методът е пределно прост – в кадъра, в който е открит преход, изходната фаза на каналите-върхове се прави равна на входната. За да има нормална фазоразвивка, това не бива да се повтаря често. Затова след открит преход се забранява откриването на следващите преходи за n следващи кадъра, като n зависи от дължината на „скока“ на припокриване на вокодера h. За използваните $h = 1024$ и $n = 4$ все още не се увеличава броят на неоткритите преходи.

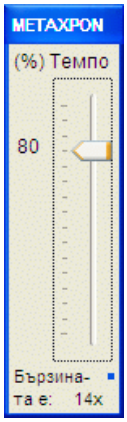
За съжаление, „замразяването“ на фазата само по себе си не осигурява достатъчно добро субективно възприятие на преходите. Поради сравнително дългия кадър, те все пак звучат леко подтиснати. За подобряване на възприемането им на слух, те трябва да се *усилят*. Това става с леко увеличаване на изходната амплитуда (около $\sqrt{2}$ пъти) в кадъра, в който е открит преходът. Съчетанието между „замразяване“ на фазата и усилване на амплитудата вече решава задачата за обработка на преходите достатъчно добре за професионални нужди.

4. Приложение, изпитания и резултати

Предложеният метод за откриване и обработка на преходи бе приложен в библиотеката за времево мащабиране на

звукови сигнали в реално време „Μεταχρον“, по-точно във функциите на фазовия вокодер с твърдо блокиране на фазата, с който се прави цифровата обработка на сигнала и́.

С цел възможно най-широкомащабни изпитания на работата и́ в реално време и върху реален звуков материал, бе написан модул за „Winamp“, в който бе използвана тази библиотека заедно с предложения метод.



Фиг. 1. Модул за ЦОС към „Winamp“ (PIII/1,4)

Англоезичният вариант на показания на фиг. 1 модул бе публикуван на сървъра на „Winamp“ под името „Προκρουστης“. За година и два месеца той е изтеглян оттам над 32000 пъти и има усреднена оценка (образувана от оценките на персонала на фирмата „Нулсофт“ – автори на „Winamp“ и на потребителите, пожелали да го оценят) равна на 4S по петобалната система. Отзивите могат да се прочетат на страницата на „Winamp“ – вж. модул („plug-in“) №147552.

Чрез измервания на бързодействието преди и след въвеждането му бе потвърдено също, че предложеният метод не внася забележимо забавяне в работата на фазовия вокодер с твърдо блокиране на фазата и следователно може да се използва за работа в реално време дори при бавен процесор.

5. Заключение

Предложеният метод за откриване и обработка на преходите е прост и ефективен и до голяма степен решава проблема с „размазването“ на преходите от фазовия вокодер без почти никакво усложняване или забавяне. По този начин става възможно мащабирането на звука във времето в реално време с достатъчно високо качество за професионални приложения, и то при такова ниско натоварване на процесора, каквото до момента не е постигано от нито един алгоритъм със сравнимо качество.

6. Благодарности

Без неоценимото съдействие на студентите от ТУ – Варна Свилен Стоянов, Станислав Кирилов и Николай Милев, италианските дисководещи Фулвио и Даниеле Ромити, както и на американската фирма „Нулсофт“, разработката и изпитанията на предложения метод биха били невъзможни!

8. Литература

[1] Flanagan, J. L. and Golden, R. M., “Phase Vocoder”, *Bell System Technical Journal*, pp. 1493–1509, 1966.

[2] Laroche, J. and Dolson, M., “Improved Phase Vocoder Time-Scale Modification of Audio”, *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, May 1999.

[3] Masri, P., “Computer Modelling of Sound for Transformation and Synthesis of Musical Signals”, *PhD Thesis*, University of Bristol, 1996.

[4] Puckette, M. S., Apel, T., and Zicarelli, D. D., “Real-time audio analysis for Pd and MSP”, *Proceedings of the International Computer Music Conference*, 1998.

[5] Bonada, J. “Automatic technique in frequency domain for near-lossles time-scale modification of audio”, *Proceedings of the International Computer Music Conference*, Berlin 2000.

[6] Rodet, X. and Jaillet, F., “Detection and modeling of fast attack transients”, *Proceedings of the International Computer Music Conference*, pp. 30–33, La Habana, Cuba, 2001.

[7] Hammer, F., “Time-scale Modification using the Phase Vocoder – An approach based on deterministic / stochastic component separation in frequency domain”, *Diploma Thesis*, Institute for Electronic Music and Acoustics (IEM), Graz University of Music and Dramatic Arts, Graz, Цsterreich, 2001.

[8] Rцbel, A., “Transient detection and preservation in the phase vocoder”, *Proceedings of the International Computer Music Conference*, pp. 247–250, Singapore 2003.

[9] Rцbel, A., “A new approach to transient processing in the phase vocoder”, *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, pp. 344–349, London 2003.

За контакти
гл. ас. инж. Лъчезар Ил. Георгиев
катедра „Компютърни науки и технологии“
Технически университет – Варна
E-mail: lucho@gawab.com

ИЗСЛЕДВАНЕ И ОПТИМИЗИРАНЕ НА АЛГОРИТМИ ЗА VOIP СИГНАЛИЗАЦИЯ

Илия Т. Танчев, Илия И. Атанасов, Розалина С. Димова

Резюме: В статията се дискутират проблеми относно работата на алгоритми за VoIP сигнализация като на пратиха демонстрират някои проблеми във VoIP тракта.Извежда се на практика работата на сигнализациите H.323 и SIP, като за целта са изследвани медийни потоци през безжична мрежа, които са софтуерно генерирани и използват едни от най-често срещаните кодеци – G.711, G.729, G.723.1 и др., и освен това каналът е натоварен с реален разговорен трафик. Информацията която се получава включва джитер, закъснения на пакетите, широчина на лентата, брой предадени пакети за изследвания интервал, закъсненията свързани със сигнализацията, разпределение на пакетите по протоколи и относителни оценки за QoS за изследваните алгоритми.

Optimize and investigate the algorithms for VoIP signalization

Ilya I. Tanchev Ilya I. Atanasov Rozalina S. Dimova

Abstract: This paper discus the problem whit algorithms for VoIP signalization end work by communication tract in normal traffic. Paper offered optimization by algorithms signalization in H.323 and SIP. For that purpose whit publication is investigate the multimedia flux by wireless lan and used very often and popular codec G.711, G729, G.723.1, ect. - software generate. Data and results included jitter, packet delay, breadth fm, numbers packets transmitted by sections, delay by signalizations, packets separation by protocols and QoS.

1. Увод

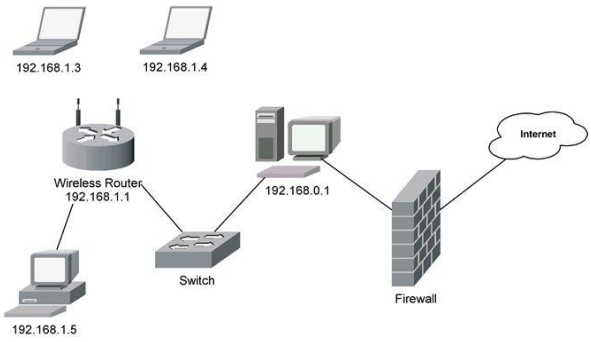
Сигнализацията на повикването при различните ситуации може да съдържа различен брой заявки и отговори, в зависимост от скоростта на сигнализацията, готовността на отсрещната страна да приеме повикването, продължителността на разговора и други фактори. [1], [2].

2.Сигнализация на повикването при SIP

Сигнализация при изграждане на директен канал

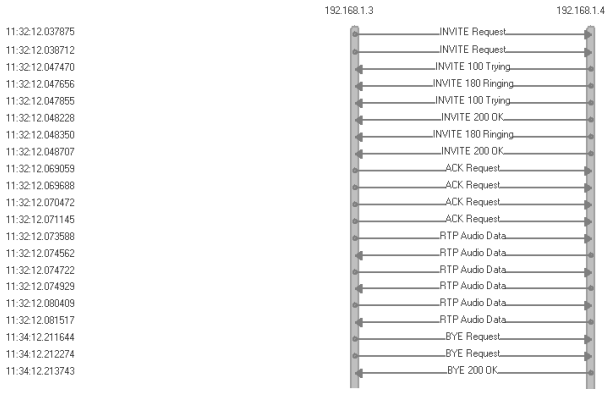
На Фиг.1.1 е показана конфигурацията на използваната мрежа за изследване на SIP трафика.

На Фиг.1.2 е показана сигнализацията при тестването на G.728 кодека, като тя е сходна и при останалите. В лявата част на фигурата се виждат моментите във времето, в които са извършени съответните заявки и са получени отговорите. Двата IP адреса 192.168.1.3 и 192.168.1.4 са съответно на викащия и викания, като те са локални за сегмента на безжичната мрежа.



Фиг.1.1 Конфигурация на мрежата

- **INVITE Request (**напълно еднакви са за двата момента от време 11:32:12.037875 и 11:32:12.038712 – това условия важи и за другите заявки и отговори) От тази първа заявка се вижда

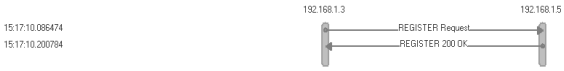


Фиг.1.2 Сигнализация при локално повикване

че потребителя към който е отправена тя има URI **sip:20001 @192.168.1.4**, за транспорт ще се използва UDP SIP Версия 2.0, повикването ще стане през **192.168.1.3 :5060**. **Branch** параметъра идентифицира дадената транзакция
Max-forwards, в случая 70, означава, че максималния брой шлюзове и прокси сървъри, през които може да премине заявката е 70.
Tag параметъра се добавя от софтуерния телефон и има идентификационни цели.
Заявката е изпратена от **sip:10001@192.-168.1.3:5060**, към **sip:20001@192. 168.1.4 :5060**, **"WinSIP2"** и **"Device 2"** са имената на източниците.
Call-ID представлява уникален идентификатор на даденото повикване, който се генерира от псевдослучайни поредици, името на хоста и IP адреса на хоста.
CSeq (Command Sequence) съдържа цяло число и името на заявката, като с всяка нова заявка това число се увеличава.
Contact съдържа *sip* или *sips* URI и пред-ставлява директния път за достигане до търсеното име.
Всички полета **Content** представляват описание на тялото на съобщението.
Accept полетата въпреки, че са допълнителни, трябва да бъдат включени тъй като те дават описание на формата на на медийните възможности.
v=0 – означава, че използваната версия на SDP е 0, което няма никакво значение, защото SIP може да работи със всички версии.
o – набор от идентификатори показващи потребителя 10001, времевия отпечатък 41532031, използваната версия на IP (IPv4) и IP адреса на викащия.
s – име на сесията, в случая WinSIP Media.
i – име на медията, Media Data.
c – информация за връзката, IPv4, IP адрес 192.168.1.3.
t=0 0 – времето в което сесията е активна.
m – името на медийния протокол и тран-спортния адрес, audio 40004 RTP/AVP 15 101.
a – показва идентификатора на профила (RTP), кодека (G.728) и честотата на дис-кретизация (8000), които трябва да се из-ползват в медийния канал

2.1 SIP сигнализация при прокси сървър

В предишната секция беше показана сигнализация на директно повикване (локално повикване), т.е. в случаите, кога-то те са в един и същ домейн. Ако край-ните точки се намират в различни домейни то тогава се налага използването на SIP прокси сървър, който маршрутизира INVITE заявките между тях. На Фиг.1.3 е показана регистрация в прокси сървър.



Фиг.1.3 Регистрация в прокси сървър

Самата сигнализация през прокси сървъра е доста близка до тази при директното повикване, като основните разлики са в полето Via. Всичко друго протича аналогично на обяснения начин.

Както се вижда от горните примери, SIP сигнализацията е обемиста, с дълги съобщения съдържащи много информация, но за сметка на това тя е лесно разбираема и дори хора без специални

познания с малко усилия биха се справили с нейното разчитане. Това е особено важна характеристика за съвременните технологии, които навлизат на пазара – да бъдат „приятелски" настроени към потребителите, което спомага за самостоятелното разрешаване на някои дребни проблеми, които възникват в процеса на експлоатация.

Цялата тази претрупаност на SIP не представлява пречка за интегрирането му в света на VoIP сигнализацията и той се приема много добре, както от IT специалистите така и от потребителите.

3. Сигнализация на повикване при H.323

H.323 е първият протокол за сигнали-зация който е бил използван в началото на VoIP ерата. Този стандарт е на ITU и описва терминалите, оборудването и услу-гите за мултимедийни комуникации през локални мрежи, които не предоставят гарантирано качество на обслужване. H.323 терминалите и оборудването могат да пренасят в реално време глас, данни и видео, както и всяка комбинация от предходните.

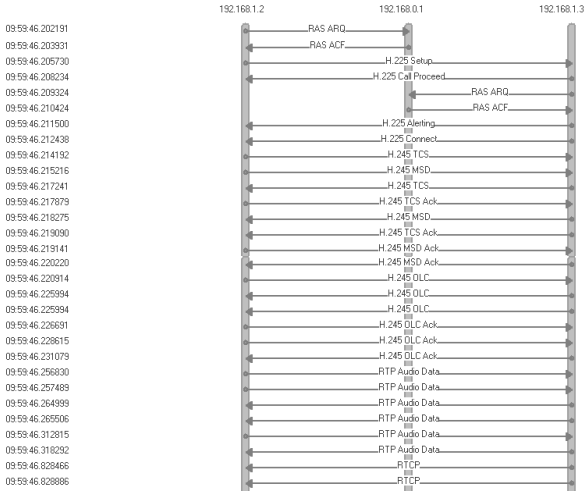
H.323 терминалите могат да бъдат персонални компютри или самостоятелни устройства като видеотелефони. Основна-та поддръжка на този протокол е за глас, докато видеото е допълнително, като и в двата случая трябва да отговарят на определени изисквания спрямо режимите за работа. Тази препоръка позволява в даден момент да се използва повече от един канал за всеки тип. H.323 включва в себе си още препоръките H.225.0 пакетиране и синхронизация, H.245 за контрол, H.261 и H.263 видео кодеци, G.711, G.722, G.728, G.729 и G.723 аудио кодеци, а също така и серията T.120 мултимедийни комуникационни протоко-ли.

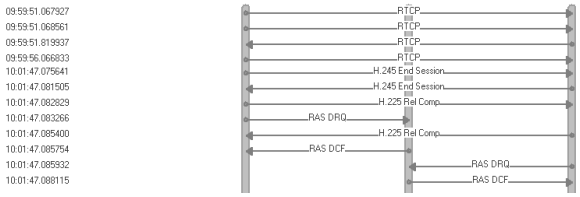
Препоръката използва сигнализационните процедури за логически канали по H.245, по която съдържанието на всеки канал се описва след като той е бил отворен. Тези процедури са необходими за описание на възможностите на предавателя и приемника, така че предавателя бива ограничен спрямо това какво може да декодира приемника, а приемника от своя страна може да заяви определен режим за предаване. Препоръката се използва и в други мрежи, например ATM, и съответно H.323 терминалите могат да взаимодействат с H.310 и H.321 терминали на B-ISDN, H.320 с N-ISDN и др. Именно тази съвместна работа с по-старите системи за комуникации дават някои предимства на H.323 пред SIP, който е напълно лишен от възможности за връзка с PSTN.

Както ще се види в процеса на анализирането при H.323 се използват много, но сравнително кратки съобщения в резултат на което сигнализацията е бърза, но за сметка на това е трудно разбираема, най-вече заради многото препоръки на ITU които са включени в нея.

Архитектурата на тестовата мрежа е аналогична на показаната на фиг. 1.1.

На Фиг.1.4 е показана реализираната сигнализация при използването на H.323. Отново както и при SIP в лявата част са показани моментите от времето, в които са изпращани заявките или съответно са по-лучавани отговорите. 192.168.1.2 е IP адресът на викащия, а 192.168.1.3 на викания. 192. 168.0.1 е адресът на входната точка (gatekeeper).





Фиг.1.4 Сигнализация при H.323.

ARQ съобщението се изпраща от крайната точка, за да се позволи нейният достъп до мрежата от входната точка, на което тя отговаря с потвърждение **ACF** или с отказ **ARJ.**

requestSeqNum – това е монотонно нарастващ номер уникален за изпращача. Той трябва да бъде върнат от приемника с всяко съобщение свързано с него.

CallType – използвайки тази стойност, входната точка може да се опита да опре-дели реалната използвана широчина на лентата. Стандартната стойност е pointTo Point, но може да се променя динамично по време на повикването.

callModel – има два варианта, при direct крайната точка заявява директен модел на повикване от терминал до терминал. При gatekeeperRouted, крайната точка заявява gatekeeper-а да бъде междинна точка, с което той не е задължен да се съобрази.

endpointIdentifier – това е идентификатор на крайната точка който се назначава на терминала от RCF и може да е E.164 адрес или H.323 ID. Използва се като мярка за сигурност, че терминала е регистриран в дадената зона.

destinationInfo – това е последователност от външни адреси за викане терминали, могат да са E.164 адреси или H.323 ID.

srcInfo – същото като при викания, само че за викащия терминал.

srcCallSignalAddress – транспортния адрес използван от източника за сигнали-зация на повикването.

bandWidth – това е позволената максимална пропускателна лента за повикване-то, може да бъде по-малка от заявената.

callReferenceValue – взема се от Q.931 за даденото повикване, има само локална валидност. Използва се за да може gate-keeper-а да асоциира дадено ARQ с опреде-лено повикване.

conferenceID – уникален конферентен идентификатор.

- **RAS ACF (192.168.0.1 -> 192.168.1.2)**

Това е потвърждението изпратено от gatekeeper-а включващо същия номер на последователността 39201, разрешената широчина на лентата и т.н.

destCallSignalAddress – в случая е транс-портния адрес към който трябва да се изпрати Q.931 сигнализацията, може да бъде gatekeeper или както в случая крайната точка (192.168.1.3 на порт 1070), като това зависи от модела на повик-ването.

irrFrequency – честотата в секунди, с която крайната точка трябва да изпраща IRR към getakeeper-а докато трае повикването или докато то се задържа.

- **H.245 TCS (192.168.1.2 -> 192.168.1.3)**

След изпращането на H.225 Connect започват процедури по проткол H.245. H.245 TCS (terminalCapabilitySet) е процедура за установяване на възможнос-тите на терминалите, с която се осигурява, че мултимедийните сигнали, които се предават ще могат да бъдат приети и обработени от приемника. От това следва и изискването всеки терминал да знае възможнос-тите на другия.

Възжможностите на предавателя описват терминалните способности за предаване на информационни потоци. Те предоставят на приемника възможност за избор на режим на работа, така че той може да заяви от своя страна режима който предпочита. Лиспата на това съоб-щение означава, че терминала не предлага възможност за избор от предпочитаните режими.

Този набор от възможности предоставя способност за едновременно предаване на информация за различните типове медийни потоци по преносната среда. Например терминала може да заяви възможността за едновременно

независи-мо предаване и приемане на видео потоци по H.262 и аудио потоци по G.722. Това съобщение също така предоставя инфор-мация за това, че терминала не е с фиксирани възмож-ности, а че са зависещи от едновремен-ните режими на работа. Например може да се индицира, че има възможност за декодиране на видео с по-висока резолюция, ако се използват по-прости аудио алгоритми или че може да се пуснат два видео потока с ниска резолюция, за сметката на един с висока.

Това съобщение също предоста-вя вариант за индициране на нестан-дартни възможности с полето Non Standard Parameter. Терминалите могат да изискват повторно договаряне на параметрите по всяко време.

- **H.245 MSD (192.168.1.2 -> 192.168.1.3)**

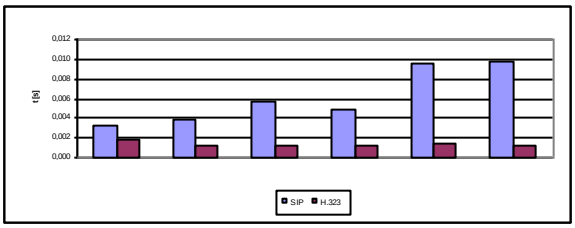
H.245 MSD (masterSlaveDetermination) съобщението се изпраща, за да се установят взаимоотношенията master-slave с което да се избегнат конфликти поро-дени от едновременна инициализация на дадени събития от два терминала в процес на повикване. Има определени правила, които дефинират как master и slave терминалите трябва да действат при въз-никване на конфликт. След като веднъж тези отношения се установят, те остават непроменени по време на по-викването

4. Анализ параметрите на VoIP трафик при сигнализации SIP и H.323

Сравняването на параметрите на два толкова различни по своето естество протоколи като SIP и H.323 е доста сложно. [3]. На първо място може да се изтъкне, че H.323 като разработка на ITU е направен с добро разбиране на изисквани-ята за мултимедийни кому-никации през пакетно базирани мрежи, включващи аудио и видео. Той опре-деля собствена система за осъществя-ване на тези функции, като взаимства известна част от вече използваните и доказали се решения в PSTN. SIP за разлика от него е проектиран за изграждане на сесии между точки, като същевременно да бъде със структура, която може да се надгражда и да се вписва гъвкаво в условията на Интернет. Резулта-та от цялата работа на проектантите е, че H.323 осигурява много добра свързаност със старите телефонни системи, благода-рение на шлюзовете, а SIP е напълно неспособен да комуникира с тях, и обрат-ното – в условията на пакетно комутирани мрежи SIP работи безупречно, докато H.323 страда от нуждата за обмен на големи обеми служебна информация. Дру-га основна разлика е, че SIP съобщенията са форматирани като текст, което помага в голяма степен за лесното им възприемане, докато при H.323 двоичния формат услож-нява нещата наистина много. Ако може да се направи кратка дефиниция на двата основни сигнализационни протока при съвременните VoIP комуникации, то тряб-ва да е, че всеки от тях е добър, стига да се използва за подходящите цели.

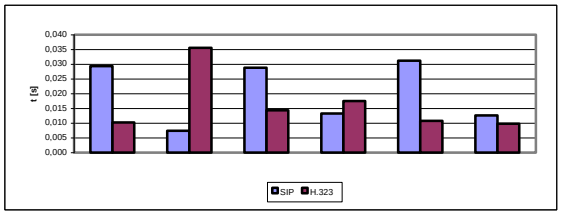
На Фиг.1.5 е показано сравнение на времената необходими на викащата страна да изпрати първия си отговор към виканата страна. При SIP това е времето между INVITE и 100 Trying, а при H.323 е между RAS ARQ и RAS ACF. Има очевидна тенденция времето необходимо на SIP да бъде по-голямо от това при H.323, като диапазона в първия случай е между 3 и 10ms, а при H.323 то е по 2ms. Това се дължи на факта, че при SIP големината на двете съобщенията е общо 1297 байта, а при H.323 – 223 байта и е нужно по-голямо време за предаването им. При SIP големи-ната може да варира в малки граници, но съотношението остава почти непроменено.

На Фиг.1.6 е сравнено времето необ-ходимо при свързването. То има по-скоро случаен характер, като стойнос-



Фиг.1.5 Initial Response Time.

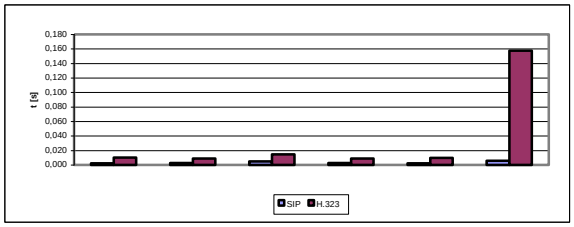
тите варират в интервала между 10 и 35ms.



Фиг.1.6 Time To Connect.

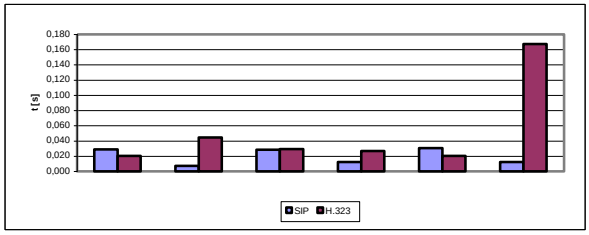
При SIP това е времето между изпращането на INVITE Request и първия ACK Request, които се изпращат от викащия към регистратора, а при H.323 е времето между RAS ARQ на викащия до момента на изпращане на H.225 Connect от викания.

На Фиг.1.7 са показани времената нужни за разпадане на повикването. При SIP това е времето между изпращането на заявката BYE Request и получаването от отсрещна-та страна на отговора BYE 200 OK. При H.323 това е времето между изпращането на първото съобщение H.245 End Session и изпращането от страна на gatekeeper-а на последното съобщение RAS DCF към ед-ната от двете страни. По големите стойно-сти при H.323 до известна степен се дължат именно на нуждата да се уведоми gatekeeper-а за освобождаването на използваната лента.



Фиг.1.7 Teardown Time.

На Фиг.1.8 са сравнени времената нужни на сигнализацията да обслужат повиква-нията. При SIP това е равно на времето на свързване (Time To Connect) без продъл-жителността на повикването (Ring Duration), докато при H.323 включва времената за свързване (Time To Connect) и тези за разпадане на повикването (Teardown Time). Преобладават времена под 50ms с изключение на последния кодек, при който се е получило близо 170ms в резултат на голямото време на разпадане.



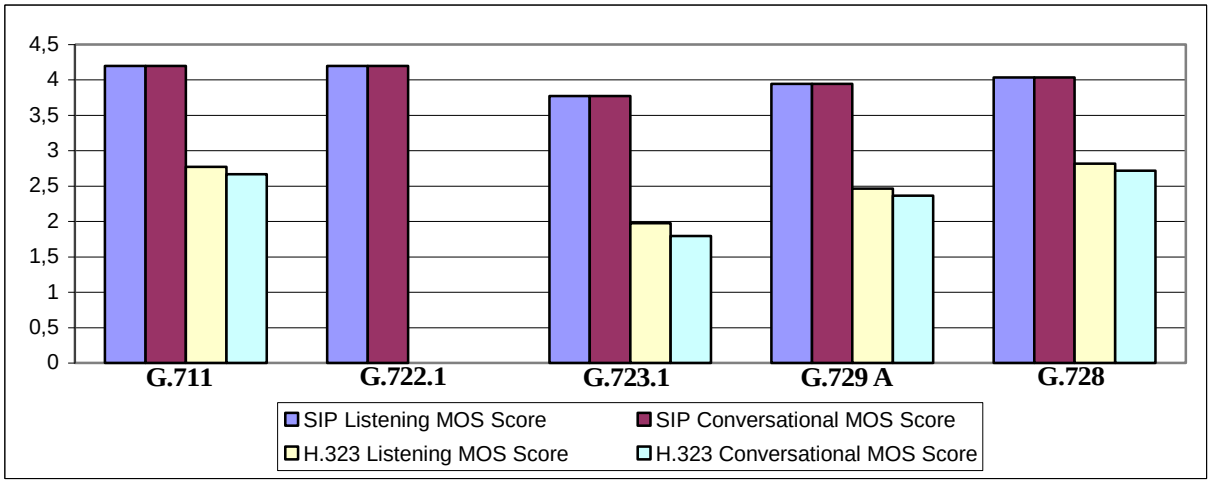
Фиг.1.8 Signaling Latency.

Най-важния параметър за една модер-на комуникационна система, каквато несъмнено е VoIP, е качеството на обслуж-ване. Докато работата свързана с проектирането на мрежата, използването на един или друг протокол, залагането на едни или други параметри и тяхното постигане е въпрос на инженерите по комуникации, то масовите потребители въобще не биха се заинтересували от това. За тях е важно услугата, която получават да е с добро качество, да е евтина, да притежава много възможности за предла-гането на нови и нови услуги.

Затова правилната оценка на способнос-тите на VoIP трябва да се направи на базата на горните критерии.

5. Заключение

На база изнесените резултати може да се отбележат следните важни моменти. На първо място MOS като техника за измерване на QoS предпо-лага работата на хора, които да дават оценки, докато в проведените изследва-



Фиг.1.9 MOS резултати от източника към получателя.

ния се използват резултатите от софтуера за анализ на VoIP трафик. Той работи по определени модели, които не винаги успяват правилно да оценят условията за провеждане на комуникацията.

Второто нещо е, че не са използвани данните от изследването на кодека AMR, който поради динамичния си характер не дава възможност на софтуера да оцени QoS, тъй като вариациите в параметрите биха довели до неправилно им тълкуване.

На Фиг.1.9 са показани MOS резултатите на изследваните кодеци в посока от източника към получателя. Тези изследвания следва да се направят в пълния обем за конкретния тракт, на базата на които авторите си поставят като задача в близко бъдеще да публикуват и алгоритъм за оптимизация.

Определено по-доброто представя-не на SIP се дължи най-вече на факта, че се използват по-малко на брой фреймове на пакет в резултат, на което интервала на пристигане на пакетите е доста по-малък от този при H.323. Особено фрапантен пример затова е G.723.1, като при SIP се пакетират по три фрейма от по 30ms, докато при H.323 се пакетират по 6 фрейма и се получава доста голям пакетен интервал. В резултат на това при всички изследвания H.323 се представи по-лошо, поне що се отнася до софтуерните MOS резултати.

Благодарности

Изследванията и резултатите изнесени в статията са изцяло финансирани по договор BY-TH 105 – фонд научни изследвания на МОН.

Литература

[1] IEEE 802.11b-1999 (R2003).
[2] **“**802.11® Wireless Networks The Definitive Guide”, M. Gast, O’Reilly 2005.
[3] “Switching to VoIP”, Theodore Wallingford, O’Reilly 2005.
[4] “Newton’s Telecom Dictionary” 16th ed., Harry Newton, CA: CMP Books.
[5] “Softswitch: Architecture for VoIP”, F. Ohrtman, McGraw-Hill.
[6] “Lab Report-QoS Solutions”, Mier Communications.
[7] “Quality of Service (QoS) in the New Public Network Architecture”, A. Agarwal, IEEE Canadian Review.

За контакти:
гл. ас. инж. Илия Т. Танчев
катедра “Съобщителна техника”
Технически университет – Варна
E-mail:itta@ms3.tu-varna.acad.bg

СИНТЕЗ НА ОПТИМАЛНИ МОДАЛНИ НАБЛЮДАТЕЛИ

Цоло Т. Георгиев, Димитър Г. Генов

Резюме: Построяването на система с използването на пълния вектор на състоянието изисква информация за всички компоненти. От икономически и технически характер на непосредствено измерване подлежи само част от вектора на състоянието, а понякога и само една компонента. Това обстоятелство изисква определяне на не измеримите компоненти от вектора на състоянието, така нареченото наблюдение на състоянието. Това се осъществява с помощта на специално устройство или алгоритъм - наблюдател, задачата на който се състои във възстановяване на пълния вектор на състоянието по неговата измерена част. В случая се предлага алгоритъм за синтез на цифрови оптимални модални наблюдатели, базиращ се на методиката, която се дава в [2].

Synthesis Of Optimal Modal Observers

Tsolo T. Georgiev, Dimitar G. Genov

Abstract: Building a system using full state vector requires information for all components. But because of economic and technical reasons on direct measuring is liable only a part of the full state vector and sometimes only one component. This circumstance requires specifying of immeasurable components of the state vector, the so called observation of the state. This is realized with the help of special device or algorithm-observer, which restores the full state vector using measured part. In this case is suggested algorithm for synthesis of digital optimal modal observers, based on methods considered in [2].

1. Постановка на задачата

Даден е математическият модел на обекта на управление във вид на уравнение на състоянието и уравнение на изхода

$$\begin{aligned}\mathbf{x}(\mathrm{k}+1) &= \mathbf{A}\mathbf{x}(k+1)+\mathbf{b}\mathbf{u}(\mathrm{k}),\quad \mathbf{x}(\mathbf{0})=\mathbf{x}_0 \\ \mathbf{y}(\mathrm{k}) &= \mathbf{C}\mathbf{x}(\mathrm{k}).\end{aligned} \qquad (1)$$

В случая са известни освен матриците **A, B** и **C** , а също така и измерените непосредствено на обекта входни и изходни сигнали. Необходимо е да се построи наблюдател на състоянието, който дава оценка $\hat{\mathbf{x}}(\mathrm{k})$ на променливата на състоянията $\mathbf{x}(\mathrm{k})$.

В случая вместо обекта (1) с обратна връзка $\mathrm{u}(\mathrm{k}) = \mathbf{K}\mathbf{x}(\mathrm{k})$ се въвежда за определяне полюсите на наблюдателя "транспонирания допълнителен обект" [3]

$$\boldsymbol{\alpha}(k+1) = \mathbf{A}^{\mathrm{T}}\boldsymbol{\alpha}(k) + \mathbf{C}^{\mathrm{T}}\boldsymbol{\beta}(k) \qquad (2)$$

с обратна връзка

$$\boldsymbol{\beta}(\kappa) = \mathbf{H}^{\mathrm{T}}\boldsymbol{\alpha}\,(\mathrm{к}), \qquad (3)$$

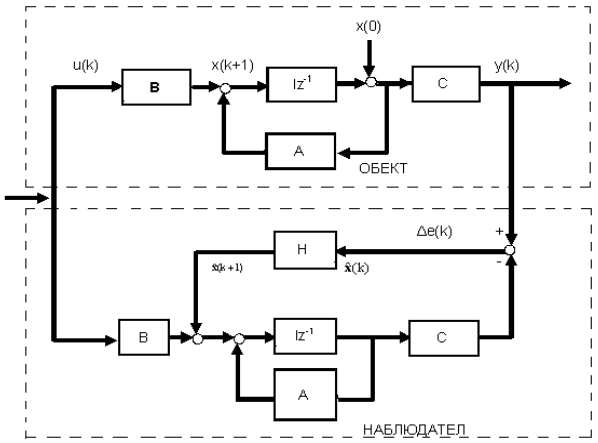
за който могат да бъдат изпълнени уравненията за синтеза на регулатора на състояние. Матрицата **H** на наблюдателя може да се определи като се използват различните методи за синтез на регулатори на състоянието. В случая за синтеза на наблюдатели на състоянията се използва методиката за оптимално модално управление разгледан в [2].

2. Алгоритъм за синтез на оптимални модални наблюдатели

На фиг. 1 е дадена схемата на обекта на управление и наблюдател на състоянието от пълен ред. В схемата е предвидена автоматична минимизация на отклонението на оценката $\hat{\mathbf{x}}(\mathrm{k})$ от нейната истинска стойност $\mathbf{x}(\mathrm{k})$, за сметка на това, че наблюдателя на

състоянието е построен като затворена система с отрицателна обратна връзка. Изхода $\mathbf{y}(k)=\mathbf{C}\mathbf{x}(k)$ се сравнява с изхода на наблюдателя $\hat{\mathbf{y}}(k)=\mathbf{C}\,\hat{\mathbf{x}}(k)$ и тяхната разлика представлява сама по себе си сигнала на несъгласуваност, който чрез коефициентите на усилване на матрицата **H** се подава на входа на интегратора.

Проектирането на наблюдателя се състои в такъв избор на структурата и параметрите на матрицата **H** , така че грешката $\tilde{\mathbf{y}} = \mathbf{y}(k) - \hat{\mathbf{y}}(k)$, а така също и грешката $\tilde{\mathbf{x}} = \mathbf{x}(k) - \hat{\mathbf{x}}(k)$ за минимално време да клонят към нула.



Фиг.1

Синтеза на наблюдателя може да се изпълни в следната последователност[2,4]:

1. Съставя се уравнението на състоянието на обекта на управление.

2. Изчислява се матрицата на наблюдаемост Q_v .

3. Определя се ранга на матрицата Q_v**.** Ако ранга на матрицата Q_v е равен на реда на системата, описваща обекта на управление то се преминава към точка 4. В противен случай е невъзможен синтеза на наблюдателя на състояние.

4. Определяне на $\mathbf{A}_e^T$ и $\mathbf{C}_e^T$ и се записва уравнението на транспонирания допълнителен обект.

5. Определят се собствените стойности на матрицата $\mathbf{A}_e^T$ от уравнението $\det[\mathbf{A}_e^T - \mathbf{I}.\chi] = \mathbf{0}$

6. Определят се "нежелателните" собствени стойности χ_1 , χ_2 , … , χ_m на отворената система, които са разположени извън единичната окръжност или на самата окръжност.

7. Дефинират се местоположенията на корените μ_1, μ_2, … , μ_m на характеристичното уравнение на затворената система, на които ще се изместят съответно "нежелателните" собствени стойности χ_1 , χ_2 , … , χ_m на отворената система.

8. Определят се елементите на собствения вектор $\mathbf{q_i}$, решавайки следната система еднородни алгебрични уравнения

$$(\mathbf{A}_e - \mathbf{I}\chi_i).\mathbf{q}_i = \mathbf{0} \qquad (4)$$

където χ_i е една от реалните собствени стойности на матрицата $\mathbf{A_e}$ на отворената система.

9. Определя се тегловната матрица $\mathbf{Q_i}$ **,** от вида

$$\mathbf{Q}_i = \mathbf{q}_i.\mathbf{q}_i^T \qquad (5)$$

10. Определят се произведенията

$$\mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T \text{ и } \mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T.\mathbf{b}_e$$

11. Определя се тегловния коефициент r_i по израза

$$r_i = \frac{\chi_i.\mu_i}{\chi_i^2.(\mu_i^2+1) - \chi_i.\mu_i.(\chi_i^2+1)} \cdot \mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T.\mathbf{b}_e \qquad (6)$$

12. Определя се коефициента λ_i

$$\lambda_i = \frac{\mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T.\mathbf{b}_e + (\chi_i^2-1).r_i}{2.\mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T.\mathbf{b}_e} + \frac{\sqrt{[\mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T.\mathbf{b}_e + (\chi_i^2-1).r_i]^2 + 4.r_i\mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T.\mathbf{b}_e}}{2.\mathbf{b}_e^T.\mathbf{q}_i.\mathbf{q}_i^T.\mathbf{b}_e} \qquad (7)$$

13. Определя се коефициента на оптималната модална връзка

$$\gamma_i^T = \frac{\chi_{\text{l}}.\lambda_i..\mathbf{b}^{\text{T}}.\mathbf{q}_{\text{i}}.\mathbf{q}_{\text{i}}^{\text{T}}}{(r_i + \lambda_{.i}.\mathbf{b}^{\text{T}}.\mathbf{q}_{\text{i}}.\mathbf{q}_{\text{i}}^{\text{T}}.\mathbf{b})} \qquad (8)$$

14. Ако са изместени всичките собствени стойности, определени в т.6 се преминава към т.15, в противен случаи се изпълнява точка 8.

15. Определя се окончателната стойност на коефициента на оптималната модална обратна връзка γ^*

$$\gamma^* = \sum_{i=1}^{m} \gamma_i \,, \qquad (9)$$

където m е броят на изместените стойности.

16. Формира се общият коефициент **H** на цифровия наблюдател.

$$\mathbf{H}=[\gamma^*]^{\mathbf{T}} \qquad (10)$$

3. Синтез на оптимални модални наблюдатели на състояние за двумасови електромеханични обекти

Векторно – матричния модел на електромеханичен обект - двигател за постоянен ток, привеждащ в движение чрез механичен редуктор тежка платформа се получава при следните допускания:

- пренебрегват се моментите на силите на мокро триене за първата и втората маса;
- пренебрегват се моментите на силите на сухото триене за първа и втора маса;
- приема се че съпротивителния момент M_c=0.

Поведението на обекта се описва със следната система линейни уравнения:

$$\begin{aligned}
\frac{\mathrm{d}\omega_1}{\mathrm{dt}} &= \frac{\mathrm{k_e}}{\mathrm{J_1}}.\mathrm{i} - \frac{1}{\mathrm{J_1}}.\mathrm{M_{12}} \\
\frac{\mathrm{dM_{12}}}{\mathrm{dt}} &= \mathrm{C_{12}}.\omega_1 - \mathrm{C_{12}}.\omega_2 \\
\frac{\mathrm{d}\omega_2}{\mathrm{dt}} &= \frac{1}{\mathrm{J_2}}.\mathrm{M_{12}} \\
\frac{\mathrm{di}}{\mathrm{dt}} &= -\frac{1}{\mathrm{T_L}}.\mathrm{i} - \frac{\mathrm{k_e}}{\mathrm{R}.\mathrm{T_L}}.\omega_1 + \frac{1}{\mathrm{R}.\mathrm{T_L}}\mathrm{U_{П}}
\end{aligned} \qquad (11)$$

Където R и L са съответно активно и индуктивно съпротивление на котвената верига, M_{12} - еластичния момент (момент, възникващ в еластичната предавка), J_1 – инерционен момент на първата маса (електродвигателя и редуктора), J_2 – инерционния момент на втората маса (платформата), C_{12} – коефициент на мокрото триене, T_L – електромагнитна константа, $U_п$ – напрежението на захранващия преобразувател и k_e – константа на електродвигателя. Обекта е със следните параметри: T_L =0.0143 s, R=2.1Ω, J_1=0.045 kgm^2, J_2=0.104 kgm^2, k_e=0.83, C_{12}=80 Nm/s^{-1}.

Ще бъде синтезиран наблюдател, възстановяващ вектора на променливите на състоянието на обекта $\mathbf{x}^{\mathbf{T}} = [\omega_1 \quad M_{12} \quad \omega_2 \quad i]$, където ω_1-скоростта на въртене на електродвигателя, M_{12}- еластичния момент, ω_2 – скоростта на въртене на механизма, i – котвения ток на двигателя. В качеството на сигнал за управление се приема изходното напрежение на преобразувателя захранващ постояннотоковия двигател, а корекцията ще бъде направена по сигнала на ъгловата скорост на първата маса.

Елементите на векторно – матричния модел

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\
y(t) &= \mathbf{C}\mathbf{x}(t)
\end{aligned} \qquad (12)$$

приемат следния вид

$$\mathbf{A} = \begin{vmatrix} 0 & -\frac{1}{\mathrm{J_1}} & 0 & \frac{k_e}{J_1} \\ C_{12} & 0 & -C_{12} & 0 \\ 0 & \frac{1}{J_2} & 0 & 0 \\ -\frac{k_e}{RT_L} & 0 & 0 & -\frac{1}{T_L} \end{vmatrix} \qquad \mathbf{b} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{RT_L} \end{vmatrix}$$

$$\mathbf{C} = \begin{vmatrix} 1 & 0 & 0 & 0 \end{vmatrix}$$

Дискретният модел на обекта в пространството на състоянието ще бъде

$$\begin{vmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} . \begin{vmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{vmatrix} + \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{vmatrix} . u(k) \qquad (13)$$

k=0,1,2…

или

$$\mathbf{x}(k+1) = \mathbf{A_d}.\mathbf{x}(k) + \mathbf{b_d}.u(k)$$

к=0,1,2...

За определяне на матрицата $\mathbf{A_d}$ и вектора $\mathbf{b_d}$ може да се използва аналитичен метод описан в [1] или възможностите на Matlab.

За целите на синтеза на наблюдателя уравнение (13) се преобразува в следната форма [2]

$$\begin{vmatrix} x_{1e}(k+1) \\ x_{2e}(k+1) \\ x_{3e}(k+1) \\ x_{4e}(k+1) \\ x_{5e}(k+1) \end{vmatrix} = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & a_{11} & -a_{12} & -a_{13} & -a_{14} \\ 0 & -a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & -a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & -a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} . \begin{vmatrix} x_{1e}(k) \\ x_{2e}(k) \\ x_{3e}(k) \\ x_{4e}(k) \\ x_{5e}(k) \end{vmatrix} + \begin{vmatrix} 0 \\ -b_1 \\ b_2 \\ b_3 \\ b_4 \end{vmatrix} . u_e(k)$$

$$y(k) = \mathbf{C}\mathbf{x}_e(k) \qquad (14)$$

или

$$\mathbf{x_e}(k+1) = \mathbf{A_e}\mathbf{x_e}(k) + \mathbf{b_e}u_e(k),\ \mathbf{x_e}(0) = \mathbf{x_{e0}},\ k = 0,1,2,...$$

$$y(k) = \mathbf{C}\mathbf{x}_e(k)$$

където:

$x_{1e}(k)=\omega_1(k)$;

$x_{3e}(k)=M_{12}(k)- M_{12}(k-1)$;

$x_{4e}(k)=\omega_1(k)- \omega_1(k-1)$;;

$x_{5e}(k)=i(k)-i(k-1)$;

$u_e(k)=u(k)-u(k-1)$;

За конкретния пример $\mathbf{A_e}$ и $\mathbf{b_e}$ се дават със следните изрази:

$$\mathbf{A_e} = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & a_{11} & -a_{12} & -a_{13} & -a_{14} \\ 0 & -a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & -a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & -a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} =$$

$$= \begin{vmatrix} 1.0000 & 1.0000 & 0 & 0 & 0 \\ 0 & 0.9989 & 0.0222 & -0.0009 & -0.0178 \\ 0 & -0.0800 & 0.9987 & -0.0800 & 0.0007 \\ 0 & -0.0004 & 0.0096 & 0.9996 & 0.0000 \\ 0 & 0.0267 & 0.0003 & -0.0000 & 0.9322 \end{vmatrix}$$

$$\mathbf{b_e} = \begin{vmatrix} 0 \\ -0.0003 \\ 0.0000 \\ 0.0000 \\ 0.0322 \end{vmatrix}$$

Еквивалентният модел на обекта, включващ грешката на системата като променлива на състоянието (14) се използва за синтез на наблюдателя на състоянието. Синтезът на наблюдателя се реализира като се изпълни алгоритъма даден в точка 2.

Синтезът на наблюдателя на състояние се изпълнява с помощта на транс-понирания допълнителен обект

$$\boldsymbol{\alpha}(k+1) = \mathbf{A}_e^{\mathrm{T}}\boldsymbol{\alpha}(k) + \mathbf{C}^{\mathrm{T}}\beta(k) \qquad (15)$$

или

$$\begin{vmatrix} \alpha_1(k+1) \\ \alpha_2(k+1) \\ \alpha_3(k+1) \\ \alpha_4(k+1) \\ \alpha_5(k+1) \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & a_{11} & -a_{21} & -a_{31} & -a_{41} \\ 0 & -a_{12} & a_{22} & a_{32} & a_{42} \\ 0 & -a_{13} & a_{23} & a_{33} & a_{43} \\ 0 & -a_{14} & a_{24} & a_{34} & a_{44} \end{vmatrix} . \begin{vmatrix} \alpha_1(k) \\ \alpha_2(k) \\ \alpha_3(k) \\ \alpha_4(k) \\ \alpha_5(k) \end{vmatrix} + \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} . \boldsymbol{\beta}(k) \qquad (16)$$

Определят се собствените стойности на матрицата $\mathbf{A_e^T}$, като се реши следното уравнение

$$\det\left[\begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & a_{11} & -a_{21} & -a_{31} & -a_{41} \\ 0 & -a_{12} & a_{22} & a_{32} & a_{42} \\ 0 & -a_{13} & a_{23} & a_{33} & a_{43} \\ 0 & -a_{14} & a_{24} & a_{34} & a_{44} \end{vmatrix} - \begin{vmatrix} \chi & 0 & 0 & 0 & 0 \\ 0 & \chi & 0 & 0 & 0 \\ 0 & 0 & \chi & 0 & 0 \\ 0 & 0 & 0 & \chi & 0 \\ 0 & 0 & 0 & 0 & \chi \end{vmatrix} \right] = 0$$

За собствените стойности се получава:

$$\chi_1 = 1; \quad \chi_2 = 0.9969 + 0.0516i;$$

$$\chi_3 = 0.9969 - 0.0516i; \quad \chi_4 = 0.9977;$$

$$\chi_5 = 0.9379$$

В случая имаме един "нежелан" корен $\chi_1 = 1$, които трябва да бъде изместен.

Дефинират се местоположенията на корена μ_1 =0.3333 на характеристичното уравнение на затворената система, на който ще се измести "нежелателната" собствен стойност χ_1 на отворената система.

За да се определи матрицата **H** на наблюдателя е необходимо да се намерят елементите на собствения вектор $\mathbf{q_1}$, като се реши системата от еднородни алгебрични уравнения..

$$(\mathbf{A_e} - \mathbf{I}\chi_i)q_i = 0 \qquad \text{за i=1,2,3}$$

За елементите на собствения вектор $\mathbf{q_1}$ се получава

$$\mathbf{q}_1 = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

Определя се тегловната матрица $\mathbf{Q_1}$

$$\mathbf{Q}_1 = \mathbf{q}_1.\mathbf{q}_1^{\mathbf{T}} = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Определят се произведенията

$$\mathbf{b}_e^T.\mathbf{q_1}.\mathbf{q}_1^T = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \end{vmatrix}$$

$$\text{и } \mathbf{b}_e^T.\mathbf{q}_1.\mathbf{q}_1^T.\mathbf{b}_e = 1$$

Изчислява се тегловния коефициент r_1 по израза (6) и се получава r_1=0.7500.

Определя се коефициента λ_1 по формула (7) λ_1=1.50.

Определя се коефициента на оптималната модална връзка γ_1^T:

$$\gamma_1^T = \begin{vmatrix} -0.6667 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

Тъй като „нежелателните" стойности в случая е само една $\chi_1 = 1$, то стойността на коефициента на оптималната модална обратна връзка γ^* се получава

$$\boldsymbol{\gamma}^* = \boldsymbol{\gamma}_1 = \begin{vmatrix} -0.6667 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Формира се вектора на обратната връзка на наблюдателя по формула (10)

$$\mathbf{H} = \begin{vmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{vmatrix} = \begin{vmatrix} -0.6667 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

4. Структурна схема на системата с оптимален модален наблюдател на състояние

Уравнението на наблюдателя съгласно [3] има следния вид

$$\hat{\mathbf{x}}_e(\mathrm{k}+1) = \mathbf{A}_e\hat{\mathbf{x}}_e(k) + \mathbf{b_e}\mathbf{u}_e(\mathrm{k}) + \mathbf{H}\Delta\mathbf{e}(\mathrm{k}) =$$

$$= \mathbf{A}_e\hat{\mathbf{x}}(k) + \mathbf{b_e}\mathbf{u}_e(\mathrm{k}) + \mathbf{H}[\mathbf{y}(\mathrm{k}) - \mathbf{C}\hat{\mathbf{x}}(k)].$$

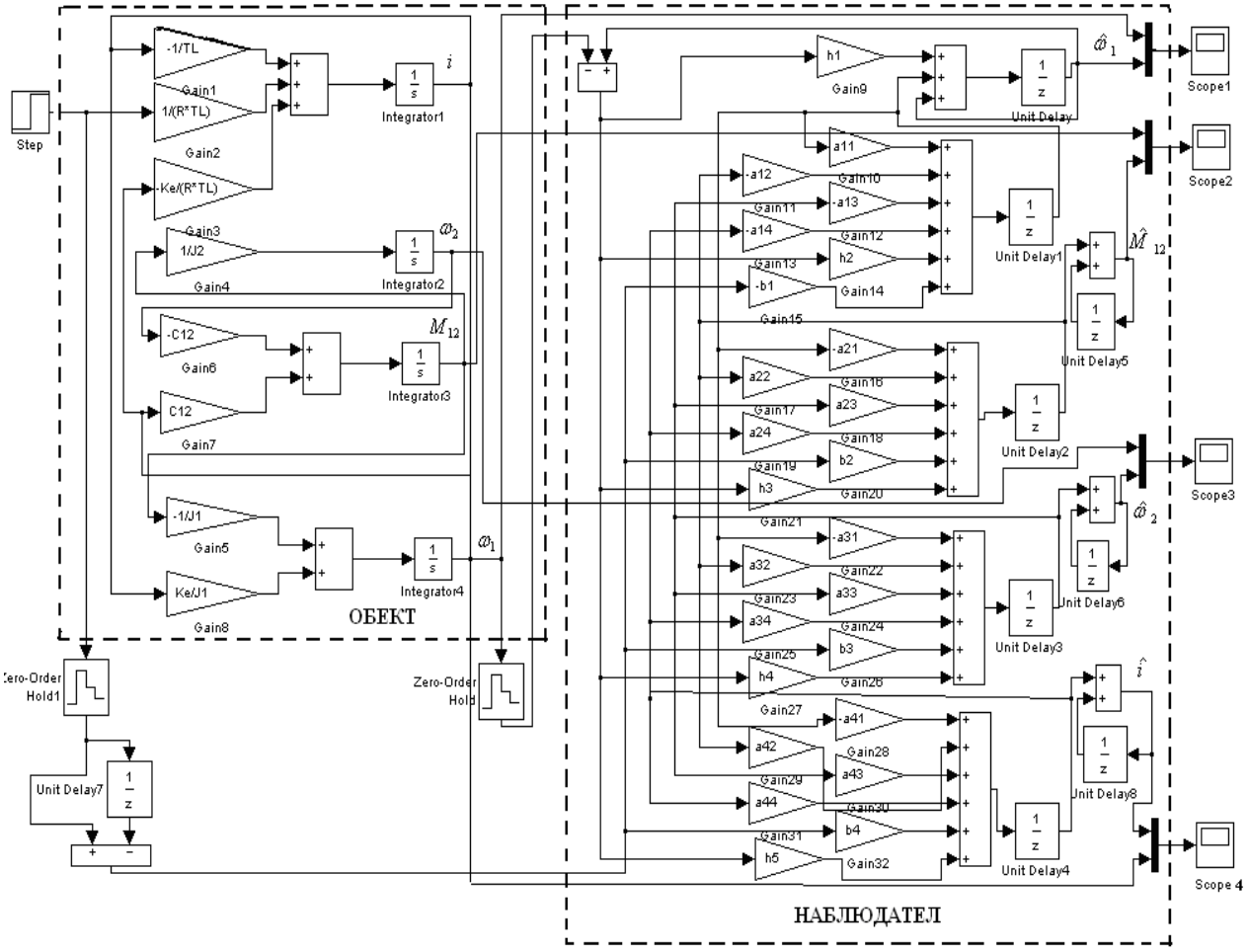
или

$$\begin{vmatrix} \hat{x}_{1e}(k+1) \\ \hat{x}_{2e}(k+1) \\ \hat{x}_{3e}(k+1) \\ \hat{x}_{4e}(k+1) \\ \hat{x}_{5e}(k+1) \end{vmatrix} = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & a_{11} & -a_{12} & -a_{13} & -a_{14} \\ 0 & -a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & -a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & -a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} . \begin{vmatrix} \hat{x}_{1e}(k) \\ \hat{x}_{2e}(k) \\ \hat{x}_{3e}(k) \\ \hat{x}_{4e}(k) \\ \hat{x}_{5e}(k) \end{vmatrix} +$$

$$+ \begin{vmatrix} 0 \\ -b_1 \\ b_2 \\ b_3 \\ b_4 \end{vmatrix} .u_e(k) + \begin{vmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{vmatrix} .\Delta e(k) \qquad (17)$$

където $\Delta e(k) = \mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}(k)$.

Скаларната форма на уравненията, с които се описва работата на наблюдателя на състояние е



Фиг. 2

$$
\begin{aligned}
\hat{x}_{1e}(k+1) &= \hat{x}_{1e} + \hat{x}_{2e} + 0.\hat{x}_{3e} + 0.\hat{x}_{4e} + 0.\hat{x}_{5e} + 0.u_e + h_1\Delta e(k)\\
\hat{x}_{2e}(k+1) &= 0.\hat{x}_{1e} + a_{11}.\hat{x}_{2e} - a_{12}\hat{x}_{3e} - a_{13}.\hat{x}_{4e} - a_{14}\hat{x}_{5e} - b_1.u_e + h_2.\Delta e(k)\\
\hat{x}_{3e}(k+1) &= 0.\hat{x}_{1e} - a_{21}.\hat{x}_{2e} + a_{22}\hat{x}_{3e} + a_{23}.\hat{x}_{4e} + a_{24}\hat{x}_{5e} + b_2.u_e + h_3.\Delta e(k)\\
\hat{x}_{4e}(k+1) &= 0.\hat{x}_{1e} - a_{31}.\hat{x}_{2e} + a_{32}\hat{x}_{3e} + a_{33}.\hat{x}_{4e} + a_{34}\hat{x}_{5e} + b_3.u_e + h_4\Delta e(k)\\
\hat{x}_{5e}(k+1) &= 0.\hat{x}_{1e} - a_{41}.\hat{x}_{2e} + a_{42}\hat{x}_{3e} + a_{43}.\hat{x}_{4e} + a_{44}\hat{x}_{5e} + b_4.u_e + h_5.\Delta e(k)
\end{aligned}
\qquad (18)
$$

Тези пет уравнения дават оценката на променливите на състоянието. Управляваща променлива, съответстваща на еквивалентния модел на обекта се дава с израза

$$u_e(k) = u(k) - u(k-1) \qquad (19)$$

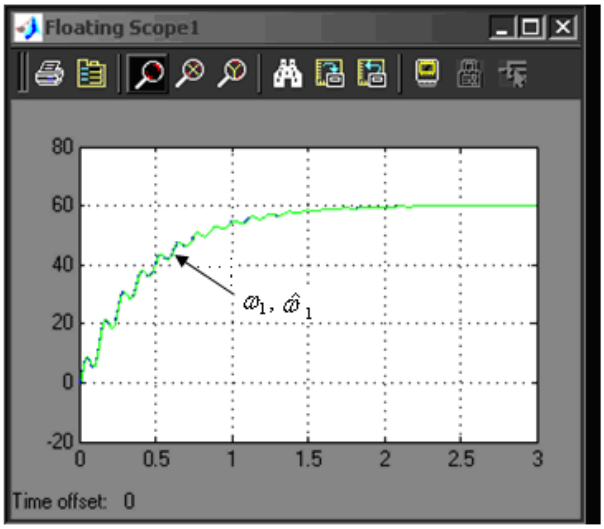
На базата на уравнения (12), (18) и (19) е разработена структурна схема на системата "обект - наблюдател на състояние". Тази структурна схема е показана на фиг. 2. На базата на тази схема е създадена програма за Simulink - Matlab. На фиг. 3, фиг. 4, фиг. 5 и фиг. 6 са представени резултатите от моделирането на системата обект – наблюдател с пакета Simulink на действителните и оценъчни характеристики съответно за

$$\omega_1 \text{ и } \hat{\omega}_1, M_{12} \text{ и } \hat{M}_{12}, \omega_2 \text{ и } \hat{\omega}_2, i \text{ и } \hat{i}\,.$$

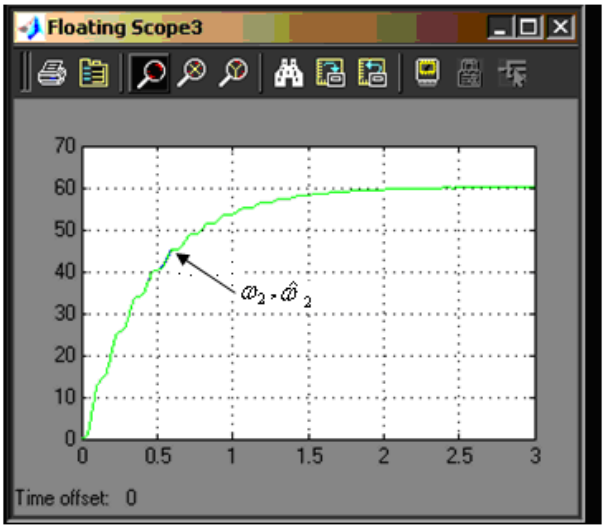
Моделиране на системата е направено при входно управляващо въздействие $U_п$=50V.

Заключение

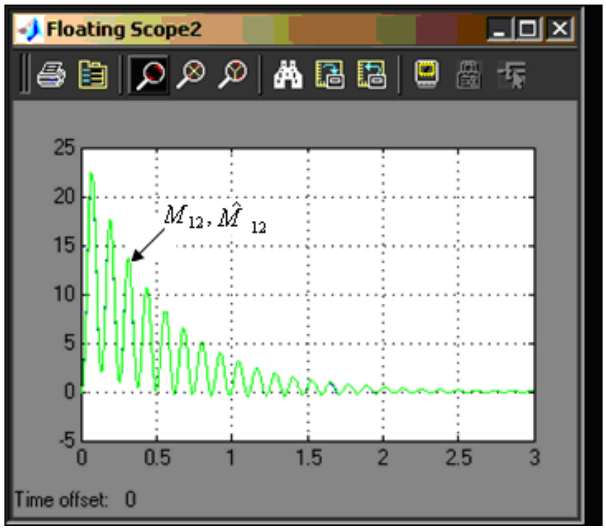
Синтезирания цифров оптимален модален наблюдател чрез измерване на управляващия сигнал $U_п$ и на сигнала на ъгловата скорост на първата маса ω_1, се възстановява цялостния вектор на променливите на състоянието на обекта $\mathbf{x}^{\mathbf{T}} = [\omega_1 \quad M_{12} \quad \omega_2 \quad i]$. Резултатите от възстановяване на пълния вектор на състоянието са дадени на фиг.3, фиг.4,



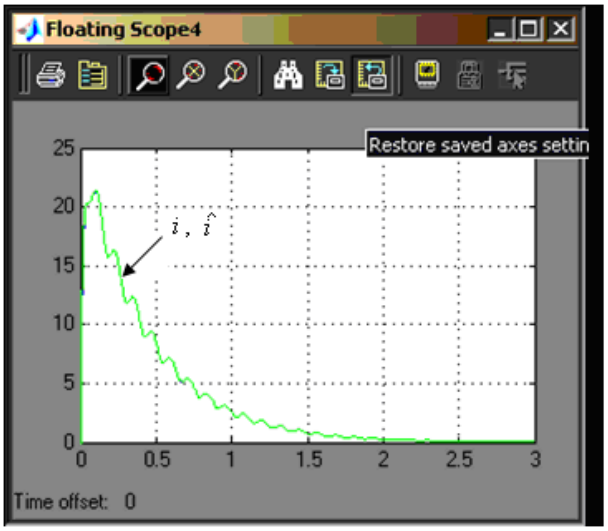
Фиг.3



Фиг.5



Фиг. 4



Фиг. 6

фиг.5 и фиг.6, от които се вижда, че оценъчните параметри

$$\hat{\omega}_1, \hat{M}_{12}, \hat{\omega}_2 \text{ и } \hat{i}$$

изцяло съвпадат с действителните

$$\omega_1, M_{12}, \omega_2 \text{ и } i \ .$$

ЛИТЕРАТУРА

[1] Барковский В.В., В.Н.Захаров, А.С. Шаталов. Методы синтеза систем управления. Машиностроение, Москва, 1981.

[2] Георгиев Ц. Т. Синтез на оптимални модални дискретни регулатори, Автоматика и информатика ‘2003 – София.

[3] Изерман Р. Цифровые системы управления. М., Мир,1984.

[4] Сотиров Л. Н. Избрани глави от съвременната теория на управлението ТУ - Варна 1998.

За контакти:

гл. ас. д-р инж. Цоло Т. Георгиев
Катедра “Автоматизация на производството”
Технически университет – Варна
E-mail: tsologeorgiev@abv.bg

доц. д-р инж. Димитър Г. Генов
Катедра “Автоматизация на производството”
Технически университет – Варна
E-mail: dggenov@yahoo.com

АВТОМАТИЧНО ГЕНЕРИРАНЕ НА УПРАЖНЕНИЯ ЗА САМООЦЕНКА В СИСТЕМА ЗА АДАПТИВНО Е-ОБУЧЕНИЕ

Бойка Ж. Градинарова, Митко М. Митев

Резюме: Статията представя подход за оценяване на знанията в средата на система за адаптивно електронно обучение, фокусирайки се върху самооценяването, явяващо се като средство за осигуряване на необходимата входна информация както за системата за адаптиране, така и за системата даваща възможност студентите да вземат решения за хода на своето обучение. Представен е модел с насочено във възходяща посока разпространение на кредитите за оценка и обратно насочено разпространение на утвърждаване и приемане на отговорите.

Automatic Generation of Exercises for Self-testing in Adaptive E-Learning Systems

Boyka Zh. Gradinarova, Mitko M. Mitev

Abstract: The paper presents aspects of knowledge assessment in the framework of adaptive e-learning systems, focusing on aspect related to self-testing, as a way of providing the input necessary for both system adaptation and user informed decisions. A model up-ward propagation of evaluation credits and down-ward propagation of validation and acceptance is presented.

Introduction

The spreading of e-learning tools usage is steadily advancing, but it is still lagging far behind expectations. E-learning does not seem to fulfil its promise to become the most important learning methodology, especially in the context of the increased role of continuous and life-long learning. This is often explained by the fact that, despite their recent impressive developments, most of the currently available e-learning tools and environments are still less appealing than the traditional face-to-face teaching methods for both students and tutors [1-2]. From the student's point of view, there are two quite opposite main reproaches: either the complaint about the "lack of human touch", the rigidity of most e-learning tools resulting in the same web pages presented to any user intending to acquire a certain item of knowledge, or the protest against the "over-coaching", the system behaving as knowing better than the user what are his/her needs. The problem of what should be part of a "learner's profile", what and how much of the learner's specific objectives, interests, preferences, and even current state of mind should be tracked as part of the concept, without rising sensitive privacy invasion issues, is still an open question. The natural answer would be to give all the control to the user, to decide how much and what type of help (s) he prefers. Still, there are cases when this approach is simply not feasible, sometimes for the mere fact that the number of parameters to set for controlling the intricacies of the behaviour of an intelligent e-learning environment is too large and full control becomes tedious and repelling in itself. From the tutor's point of view, the main drawback of current e-learning systems is that they tend to require more effort in authoring teaching materials and in preparing lessons, tests and examinations than their classical counterparts, while effectively isolating the student by hiding the whole educational environment, teacher and peers included, behind a machine. The capacity of largely re-using teaching materials, while still filtering them through the tutor personality to an extent

that gives the sense of recognized authorship, the permanent synchronous and asynchronous contact among learners and between learners and teachers to a level that helps establishing an effective cooperative learning environment seems to be the way to the answer in this case. Adequate authoring tools are needed to help the teachers in preparing both learning materials and tests for evaluating the advancement of the students towards their teaching goals. Traditional knowledge assessment is known to be demanding from both parts involved, time consuming, rising emotional aspects that do not contribute to the co-operation among the tutors and students and are prone to trigger a negative attitude from the learners. Nevertheless, a large variety of testing and evaluation tools are required by any intelligent e-learning environment to get the feedback necessary for the adaptation of the teaching system to guide the student along the learning process. Such tools operate best when they are perceived by the user as self-testing helps, giving him/her an effective support and remaining continuously under his/her own control. The learner's full co-operation makes the testing tools useful and an active part of the learning environment.

The paper briefly presents an ILE that implements the computer-supported collaborative work model, focussing on aspects referring to automatic authoring of the self tests for student evaluation the tools used to evaluate and guide the students' advancement towards their learning goals.. A prototype automatic generator of ac circuit exercise is presented in some details, stressing on the general points that could be used for test and exercises in any field of science and engineering involving quantitative analysis.

1. Intelligent e-Learning Systems

The vast majority of the currently used web-based educational systems are powerful integrated systems, like Blackboard [3] or WebCT [4] that provide a large variety of support services to both learners and teachers, but lack adaptability, belonging to the class of Learning Management Systems (LMS). Significant research and implementation effort has been dedicated to develop Intelligent Tutoring Systems [5] and Adaptive Hypermedia [6], able to adapt to learner's objectives, interests, and preferences, i.e., to Learner Profile (LP). Currently, such systems offer remarkable adaptability, but only for specific tasks, lacking integration. Improved distributed architectures, centred on the concept of Intelligent Learning Environments, have been suggested to allow combining the efficiency of LMS with the flexibility of adaptive systems [7-8]. To implement the adaptivity feature, an ILE requires a quite complex structure, with several parallel version of the same learning item (LI), allowing many different learning paths to be selected in accordance with the LPs. This approach requires considerable additional effort in elaborating teaching materials, might require several authors and might need institutional support, but brings the advantage of true flexibility and adaptability. A course, as a teaching material, is no longer a flat juxtaposition of learning items, but a multilevel structure with many parallel branches, along which the ILE recommends an optimal path for a user or for a class of users. As mentioned in [10], it has been argued that authoring learning material and building the structure of adaptive systems tends to become too complicated for the average teacher. Correspondingly, portability – the ability to deploy the content of a system on any other system, and reusability – the ability to search and retrieve learning items (LIs), including lessons, modules, exercises, activities and to reuse them, become strictly necessary features for an efficient implementation of the ILE concept and for the success of the wide scale implementation of this approach. The problem has already been addressed for the case of LMS: the DOD's Sharable Content Object Reference Model (SCORM) ADL initiative [10] aims to provide a comprehensive set of e-learning capabilities enabling interoperability, accessibility and reuse of Web-based learning content. Similar efforts are under way to provide a harmonized set of guidelines,

specification and standards for Intelligent Tutoring Systems (ITS) and Adaptive Hypermedia (AH) systems [11, 12]. At the same time, significant changes in the basic methodology of teaching/learning is to be expected as result of the current advancement and convergence of cognitive psychology and computing science. Paradigms such as just-in-time learning, constructivist approaches, student-centred learning and collaborative approaches have emerged, and are being supported by technological advancements including simulations, virtual reality and multi-agents systems.

2. Methodology, Architecture and Implementation

The system is learner centered, all human and artificial agents being focused on achieving the learning-training tasks. Human agents include, aside the students, authors of teaching materials, tutors, course administrators, and system administrator(s). The pilot web oriented ILE has the server-client distributed multiagent hybrid architecture shown in Figure 1 [12]. The client side requirements are kept to a minimum; each human agent accesses the system via an intranet or internet connection, by using a standard web browser. On the server side, each human agent has a personal assistant (agent)

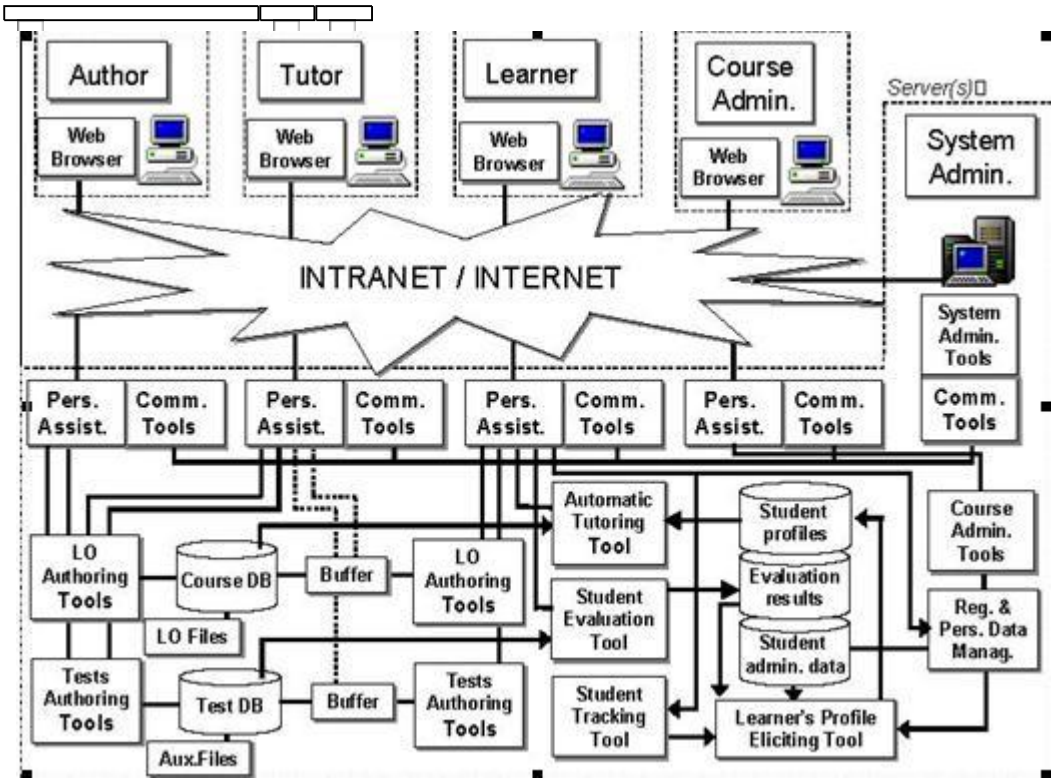


Fig. 1: Architecture of the ILE pilot

that inter-operates with the other artificial agents providing the system functionality. All personal assistants are linked to communication tools that facilitate the contact among human actors in the system, and provide support to the collaboration among the students. The learner personal assistant (LPA) controls, in the first place, the access to the agents providing the main functions – the Automatic Tutoring Tool and the Student Evaluation Tool, but

also to the Learning Item (LI) Authoring Tool and Tests Authoring Tool – thus providing support to a participative learning approach. Students get credits not only for their results in the tests, but also for their involvement in the course development – by contributing with new versions to existing *LI*s, building new questions for tests addressing various *LI*s, etc. When submitted, the student contributions are stored in a buffer, waiting for tutor's validation to be introduced in the course and test databases. The contributions are marked similarly to the successful passing of regular tests. The LPA gives also access to the Registration and Personal Data Management Tool, and helps tracking student participation and results, passing the corresponding data to the Learner's Profile Eliciting Tool (LPET). The tutor has also a Tutor Personal Assistant (TPA) that provides an interface to all system functionalities offered to a tutor: keeping track of class enrolment, following learners' progress in the training process, reading synthetic data on learners' profile, etc. Authors and tutors have access to authoring tools, to update the course, tests and any other teaching materials. Course administrators have access to all relevant data about courses and students. The system administrator monitors the usage of the resources and takes care of the smooth functioning of the system. The distribution of the functional roles among human actors can be managed from an administrator platform accessed via root privileges.

The server has been implemented as an in-house developed J2EE compatible JAVA web-application, running on the Tomcat Apache Server and using the MySQL database server, both largely accessible and available on UNIX and WINDOWS platforms. The modular approach allows an easy restructuring of the system; e.g., the change of the database affects only the module controlling the database. Database access speed is increased by using a connection pool. Data access (some contained in the MySQL database, some in a system of related directory files) is made through interactive web pages implemented with Java Server Pages (JSP) and Servlets. Each JSP is controlled by a JavaBean which handles the security tasks and data manipulation. Every JavaBean within a module extends the class corresponding to an actor (learner, author, tutor, course and system administrator), being able to control authorized access.

3. Learning Evaluation

Learning evaluation is based on test results, but also on the student involvement in proactive learning, including the contribution to the course continuous development [13]. Care is taken to stimulate students to do more than just recognize textbook material when responding to tests. Students are encouraged to formulate themselves new questions and are given credits when their contribution is included in the question data base (QDB).

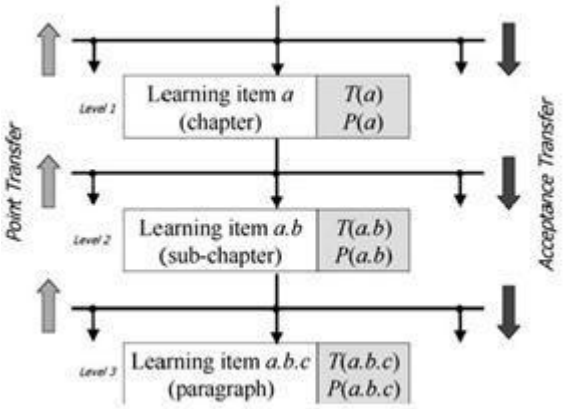


Fig.2. Evaluation points and acceptance transfers

To make the system closer to a classic examination that doesnot require the checking every item in a course, the passing of a higher level *LI* is not conditioned by the passing of all lower *LI*s it contains, but only by the sum of points exceeding the threshold for that level. The acceptance of a higher level *LI* gives credit for all *LI*s below it in the course structure. This results in an up-wards propagation of the awarded points and a downward propagation of the acceptance, as shown in Figure 2. The acceptance of an *LI* is irreversible, so that only warnings and advices are issued if later errors signal possible weaknesses in an transfers already assessed LI. The tests are built by randomly choosing from

the question data base (QDB) a specified number of questions referring to the given tag (*LI* address or keyword). A test referring to a *LI* can comprise questions attached either directly to that specific *LI*, or to any *LI* placed below it in the course tree structure. The points obtained when making a choice *C* from the set

$$SP(Q) = \sum_{C \in S(Q)} P(C) \,, \quad (1)$$

whre $S(Q) \subseteq O(Q)$ is the set of options the student has selected at question Q. The correct choices are awarded positive points, the wrong answers – negative points. Assigning negative points to the wrong choices contributes to discourage guessing, but the system success relies on student motivation. The points *P*(*Q*) acknowledged for question *Q* is given by:

$$P(Q) = \begin{cases} SP(Q), if SP(Q) < 0, \\ 0, if\ 0 \le SP(Q) < T(Q), \\ SP(Q), if SP(Q) \ge T(Q) \end{cases} \quad (2)$$

where *T*(*Q*) is the threshold required for the acceptance of the reply to *Q*. As a consequence, points obtained for a question are taken into account only when exceeding a certain requested minimum. Such a learning appraisal aims at a robust understanding and proper using of the tested knowledge, by discouraging superficial and fragmentary learning. Each *LI* is not evaluated independently, but in the context of the course, in relation with the related *LI*s. The sum of points *SP*(*LI*) for a certain learning item *LI* consists not only of the sum of the points obtained for the questions *Q* referring directly to *LI*, but, also, of the sum of the acknowledged points transferred to *LI* from its children – the learning items *LI'* placed one level below it in the course structure:

$$\mathrm{SP(LI)} = \sum_{Q \in LI} P(C) + \sum_{LI \in C(LI)} P(LI) \,, \quad (3)$$

of options O(*Q*) pertinent to question *Q* are recorded at the LI to which the question is attached and transferred upwards. The sum of points *P*(*Q*) for a question *Q* results by adding the points for all options selected at *Q*:

where *C* (*LI*) are the children of *LI*. Equation (3) ensures that the points obtained for a certain *LI* are transferred upwards, to all its ascendants, without double counting.

The points *P*(*LI*) acknowledged for a learning item *LI* depend also on the passing of a certain acceptance threshold *T*(*LI*), but, in this case, there is an award *A*(*LI*) for the successful completion of the study of *LI* marked by the passing of *T*(*LI*):

$$\mathrm{S(LI)} = \begin{cases} SP(LI), if SP(LI) < T(LI), \\ SP(LI) + A(LI), if SP(LI) \ge T(LI). \end{cases} \quad (4)$$

The status *S*(*LI*) of *LI* changes from 0 – *pending* to 1 – *studied*, when its threshold, or the threshold of its ascendant, have been passed:

$$\mathrm{S(LI)} = \begin{cases} 0, if\ (SP(LI) < T(LI)) and \\ (S(LI') = 0, LI \in C(LI')), \\ 1, if\ (SP(LI) \ge T(LI)) or \\ (S(LI' = 1, LI \in C(LI)), \end{cases} \quad (5)$$

where *C* (*LI'*) are the children of *LI'*.

Relation (3) shows the up-propagation of the points in the tree-like course structure, while relation (5) corresponds to the down-propagation of the acquired knowledge recognition along the same structure.

4. Conclusions

The paper presents a pilot implementation of an Intelligent e-learning environment (*ILE*) able to adapt to the learner's profile (*LP*) and to encourage active and cooperative learning. Special attention is

given to support the authoring of learning items and tests, especially important in an ILE context to assess the students advance towards their learning objectives. Adequate authoring tools can make the tutors task easier, contributing to a better acceptance of e-learning systems, despite the required extra work and can stimulate a pro-active attitude of students. An example of automatic problem generation is given for the case study of a.c. electric circuit analysis. A similar methodology can be applied to convert any engineering analysis problem into a procedure for automatic generation of tests. This approach has the advantage of greatly simplifying the problem of synthesizing system analysis exercises, to the point of making them available to both tutors and students, in the later case for self-testing and computational experiments purposes.

References

[1] D. A. Norman, J.D. Spohrer, Learner-centered education, Comm. of the ACM, 39(4), 1996, pp. 24-27.

[2] M. J. Jennings, Intelligent agents: Theory and practice, The Knowledge Engineering Review, Wooldrige, N.R. 10(2), 1995, pp. 115-152.

[3] Blackboard Inc, Blackboard Course Management System, http://www.blackboard.com/.

[4] WebCT, Course Management System, http://www.webct.com/.

[5] C. Frasson, G. Gauthier (editors), Intelligent Tutoring Systems - At the Cross-roads of Artificial Intelligence and Education, Ablex Publishing Corporation. Norwood, New Jersey, 1990.

[6] P. Brusilovsky, Adaptive hypermedia, User Modeling and User Adapted Interaction, Ten Year Anniversary Issue (Alfred Kobsa, ed.) 11(1/2), 2002, pp. 87-110.

[7] P**.** Brusilovsky, Student model centered architecture for intelligent learning environments, Proc. of Fourth international conference on User Modeling, 15-19 August, Hyannis, MA, USA. User Modeling Inc, 1994, pp. 31-36.

[8] O. Conlan, C. Hockemeyer, V. Wade, D.Albert, D., & M. Gargan, An architecture for integrating adaptive hypermedia service with open learning environments, In Proc. of ED-MEDIA 2002, vol. 1 of World Conference on Educational Multimedia, Hypermedia & Telecommunications, pp. 344-350.

[9] P. Brusilovsky, A Distributed Architecture for Adaptive and Intelligent Learning Management Systems, Proc. of AI in Education (AIED2003), vol.4., Workshop Towards Intelligent Learning Management Systems, Sydney, Australia, July 2003.

[10] Advanced Distributed Learning (ADL) Initiative, Sharable Content Object Reference Model (SCORM), http://www.adlnet.org/.

[11] D. Dicheva, L. Aroyo, Alexandra Cristea, Collaborative Courseware Authoring Support, Proc WEB-CATE 2002 – Computers and Advanced Technology in Education, Cancun, Mexico, 2002, pp. 52-57.

[12] P. D. Cristea, Rodica Tuduce, Pilot Implementation of an Intelligent e-Learning Environment, eChallenges e2004, Vienna, Austria, October 27-29, 2004.

For Contacts:

Boyka Zh. Gradinarova,PhD

Assoc. Prof. Mitko M. Mitev, PhD

Department of Computer Science and Tehnologies

Techical Universety of Varna.

E-mail: BGradinarova@hotmail.com

E-mail: fs_centr@ms3.tu-varna.acad.bg

Информация за Института на Инженерите по Електротехника и Електроника (IEEE)

Йордан Колев

Институтът е най-голямата техническа професионална организация в света. Има около 400 000 члена в над 150 страни. Официално е създаден през 1963 г. в резултат на сливането на Американския институт на електроинженерите (AIEE) - основан през 1884 г. и Института на Радиоинженерите (IRE) - основан през 1912 г.

Институтът съдейства за развитието на електротехниката и сродните и науки , включвайки всичко от космоса, енергетиката, транспорта до комуникациите, компютрите и микроелектрониката, като работи за прилагането на тези технологии в полза на човечеството.

Институтът има редица професионални научно-образователни и обществени цели - подкрепя професионалното развитие и колективния стил на работа, осъществяването на дългогодишна професионална кариера, стреми се да популяризира и да повиши привлекателността на електротехниката и сродните науки, и постепенно се е превърнал от американска в глобална организация, чувствителна към многообразието на националните култури и традиции. Научно-образователните си цели IEEE осъществява чрез своите публикации, конференции и образователни програми. IEEE публикува една четвърт от световната литература в областта на електротехниката и сродните и науки - около 120 периодични списания, 200 сборници от конференции, десетки книги, над 100 стандарта. Голяма част от публикациите на IEEE са достъпни в Internet и на CD ROM. Списанието с най-голям тираж, получавано от всеки член на IEEE е SPECTRUM. Там се публикуват обзорно-аналитични статии в целия спектър от професионално-технически и научни области на IEEE. С висок престиж се ползуват специализираните научно-технически списания в различните области (Transactions on....), както и обзорно-аналитичните Proceedings of the IEEE.

В провежданите всяка година от IEEE над 300 конференции участвуват повече от 350 000 инженери и учени. Освен това под егидата на IEEE се организират и близо 4500 локални срещи по целия свят.

IEEE активно работи по развитието на световните технически стандарти. Ежегодно благодарение на труда на повече от 3000 специалисти се одобряват и публикуват над 100 нови и преработени стандарти, включително промишлени и международни.

Отчитайки динамиката на научните знания и необходимостта от непрекъснато обучение IEEE издава също курсове за дистанционно обучение и учебни видеофилми, участвува активно в преразглеждането на университетските учебни планове и програми, както и в акредитацията на обучението по електротехническите и компютърните науки. Наградите присъждани от IEEE за научно-технически постижения са едни от най-престижните международни отличия.

Структурата на IEEE включва субекти разделени по географски и професионален признак.

Има 39 професионални сдружения (Societies). Първото образувано сдружение е по Обработка на сигнали (Signal Processing). Трите най-големи професионални сдружения са: Компютри, Кому-

никации и Електро-енергетика (Power Engineering).
Основният субект на географската структура на IEEE е Секцията. В света има над 300 секции, от които около 180 са в САЩ обединени в региони от 1 до 6, а над 120 са в останалите страни обединени в региони 7 до 10.
В рамките на 39-те професионални сдружения към отделните секции са изградени над 1000 професионални клона (Chapters).

Българска секция на Института на Инженерите по Електротехника и Електроника (IEEE Bulgaria Section)

Българската секция на IEEE бе утвърдена от Изълнителния Комитет на IEEE на 24 юни 1995. Създаването на Българската секция стана възможно благодарение на усилията на редица колеги, членове на IEEE, работили в продължение на години за популяризирането на тази престижна световна професионална организация и за привличането на нови членове. Утвърждаването на секцията е резултат от изпращането до Изпълнителния Комитет на петиция подписана от 57 български членове на IEEE.
Наред с това бе утвърден и първият професионален клон на секцията в областта на Микровълновата техника, Електронните прибори, Антените и разпространението на електромагнитните вълни (IEEE Bulgaria MTT/ED/AP Joint Chapter).
Целите и задачите на българските структури на IEEE са в съответствие с целите и задачите на IEEE, адаптирани към потребностите на българските инженери. По-важните от тях са: облекчаване на достъпа на българските инженери и учени до актуалната информация от публикациите, конференциите и Internet-сървериte на IEEE, увеличаване броя на членовете на IEEE в България, а също и броя на студентите-членове и създаването на студентски клонове в техническите университети, подпомагане на техническите и университетски библиотеки чрез включването им в програмата за спонсориране от IEEE, организиране на съвместна дейност с другите професионални научно-технически съюзи, провеждане в България на конференции под егидата или спонсорирани от IEEE.
В момента Българската Секция има близо 300 члена, от които около половината са студенти.
В състава на секцията действуват 9 Клона (ED-15/MTT-17/AP-03/CPMT-21, SP-01, CAS- 04/SSC-37, C-16, IM-09/CS-23/SMC-28, COM-19, ED-15/SSC-37 VARNA, RA-24, CIS-11)

За контакти:
доц. д-р инж. Йордан Колев,
Катедра "Електронна техника и микроелектроника"
Технически университет – Варна
Председател на БС на IEEE
тел/факс: 052 302759
e-mail:j.n.kolev@ieee.org

Изисквания за оформяне на статиите на авторите за списание "Компютърни науки и технологии”

I. Ръкописите на статията се представят в два екземпляра (оригинал и копие) в размер до 6 страници формат А4 и като файл на дискета, на адрес: Технически университет – Варна, катедра “Компютърни науки и технологии”, ул. Студентска 1, 9010 Варна.

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 адрес за контакти, включващ: научно звание и степен, име, инициали, фамилия, организация, поделение(катедра), e-mail адрес.

III. Всички математически формули трябва да са написани ясно и четливо (препоръчва се използуване на Microsoft Equation). Номерацията на формулите се дава вдясно от началото на реда. Текста на формулите се позиционира в средата на реда. Експоненциалните функции се изписват чрез знака exp.

1. Текстът трябва да бъде въведен във файл във формат WinWord 2000/2003 със шрифт Times New Roman. Форматирането трябва да бъде както следва:
2. Размер на листа - А4, полета - ляво - 20мм, дясно - 20мм, горно - 15мм, долно - 35мм.
3. Заглавие - (на български език, размер на шрифта 16, bold).
4. Един празен ред, размер на шрифта 14, нормален.
5. Имена на авторите - име, инициали на презиме, фамилия, без звания и научни степени, размер на шрифта 14, нормален.
6. Два празни реда, размер на шрифта 14.
7. Резюме -(на български език, размер на шрифта 11) - до 8 реда, нормален.
8. Заглавие (на английски език, размер на шрифта 12) – bold.
9. Един празен ред, размер 11.
10. Имена на авторите (на английски език, размер на шрифта 11) – нормален.
11. Един празен ред, размер 11.
12. Резюме -(на англиийски език, размер на шрифта 11) - до 8 реда, нормален.
13. Основните раздели на статията (Увод, Изложение, Заключение, Благодарности) се форматират в двуколонен текст с разстояние между колоните 10мм както следва:

n. наименование на раздел или на подраздел, размер на шрифта 12, удебелен, центриран.
o. празен ред, размер на шрифта 12;
p. текст, размер на шрифта 12, нормален, отстъп на първи ред на параграф – 10 мм;
q. цитиране на литературен източник - номер на източника от списъка в квадратни скоби;
r. литература – всеки литературен източник се представя с: номер в квадратни скоби, списък на авторите, издателство, град, година на издаване.
s. За контакти: научно звание и степен, име, инициали на презиме, фамилия, организация, поделение(катедра), e-mail адрес.

Образец за форматиране (компресиран Word файл) можете да изтеглите от адрес http://kst-forum.netfirms.com/ Spisanie_Obrazec.zip