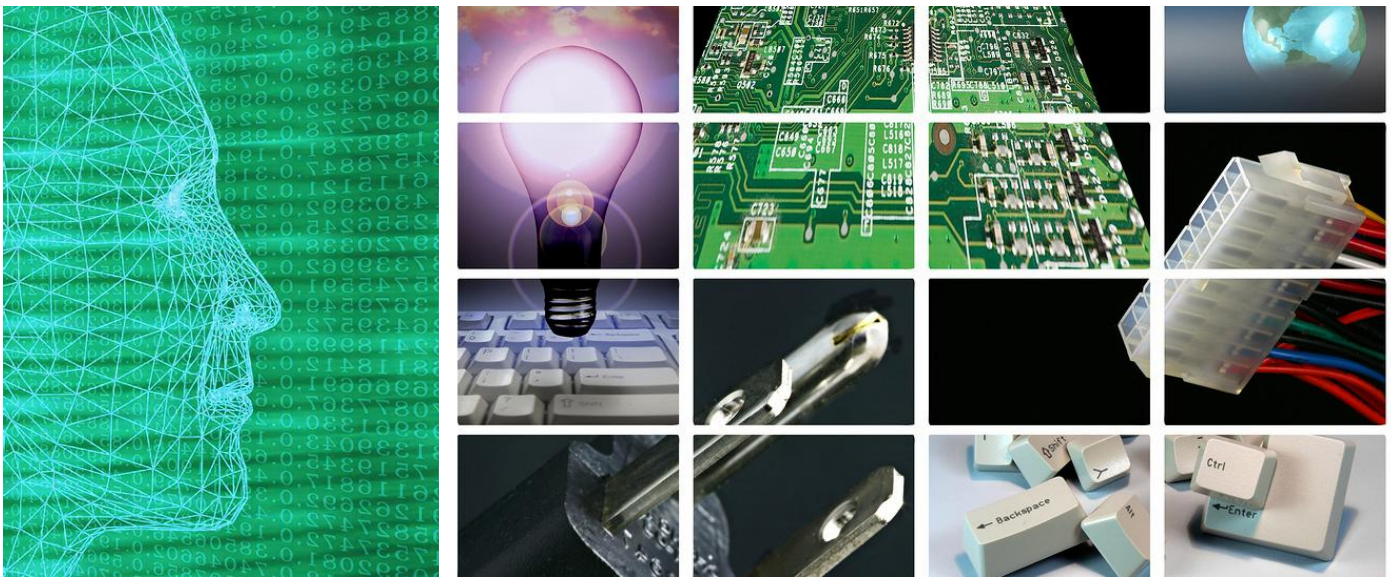
ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА
TECHNICAL UNIVERSITY OF VARNA

Година X, Брой 1/2012

КОМПЮТЪРНИ НАУКИ И ТЕХНОЛОГИИ



IEEE Bulgaria Communications Chapter

FCA Faculty of Computing & Automation

COMPUTER SCIENCE AND TECHNOLOGIES

Варна, България

Year X, Number 1/2012

Компютърни науки и технологии

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

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

Редакционна колегия:

проф. дтн. Л. Сотиров (Варна)
проф. дтн. С. Антощук (Одеса)
проф. дтн. С. Стойчев (София)
проф. д-р А. Смрикаров (Русе)
доц. д-р А. Антонов (Варна)
доц. д-р В. Наумов (Варна)
доц. д-р Г. Тодоров (В. Търново)
доц. д-р Е. Маринов (Варна)
доц. д-р Н. Рускова (Варна)
доц. д-р Р. Райчев (Габрово)

Печат: ТУ-Варна

За контакти:

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

ISSN 1312-3335

Computer Science and Technologies

Publication
of Computing and Automation Faculty
Technical University of Varna

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

Advisory Board:

Prof. L. Sotirov, DSc (Varna)
Prof. S. Antoshchuk, DSc (Odessa)
Prof. S. Stoichev, DSc (Sofia)
Prof. A. Smrikarov, PhD (Ruse)
Assoc. Prof. A. Antonov, PhD (Varna)
Assoc. Prof. V. Naumov, PhD (Varna)
Assoc. Prof. G. Todorov, PhD (V. Tarnovo)
Assoc. Prof. E. Marinov, PhD (Varna)
Assoc. Prof. N. Ruskova, PhD (Varna)
Assoc. Prof. R. Raychev, PhD (Gabrovo)

Printing: TU-Varna

For contacts:

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

ISSN 1312-3335

МУЛТИФОРМЕН, МУЛТИФОРМАТЕН ЦИФРОВ КОМПАРАТОР

Димитър С. Тянев, Димитър Г. Генов,
Веселин В. Стефанов, Мартин Д. Георгиев, Стефка И. Попова

Резюме: Аналитично е обоснована възможността за синтез на мултиформен и мултиформатен цифров компаратор, основаващ се на алгоритъма за без знаково сравнение на двоични числа. Синтезираният компаратор представлява нова оригинална логическа схема, която е способна да сравнява цели и дробни двоични числа със знак, както и двоично-десетични такива, представени в различни инверсни машинни кодове и кодирани в различни двоично-десетични кодове. Същата схема е в състояние да сравнява и числа, представени във форма с плаваща запетая според стандарта IEEE-754. Схемата е с висока степен на универсалност. Тя с успех може да замени в цифровите процесори традиционния алгоритъм за операция сравнение, реализирана чрез изваждане. Като случайна величина е изследвана латентността на компаратора. Определен е закона за разпределението й и неговите параметри.
Ключови думи: компаратор, цифров компаратор, признаци *LT, EQ, GT*, двоични числа, двоично-десетични числа, код 8421, 8421^{+3}, 2421, числа с плаваща запетая, стандарт IEEE-754, латентност, критерий на Пирсон, геометрично разпределение

Multi-form, Multi-format Digital Comparator

Dimitar S. Tyanev, Dimitar G. Genov,
Veselin V. Stefanov, Martin D. Georgiev, Stefka I. Popova

Abstract: There is analytical grounded possibility for synthesis of multi-form and multi-format digital comparator based on the algorithm for unsigned comparison of binary numbers. The synthesized comparator is a new original logic scheme capable of comparing integer and fractal signed binary numbers, as well as binary-coded decimal numbers presented in different inverse codes and various binary-decimal codes. Same scheme can compare also numbers presented in a floating point format according to IEEE-754 standard. The scheme has high degree of universality. It can replace successfully traditional algorithm for comparison used in digital processors which is implemented by subtraction. The comparator latency has been explored as a random quantity. The distribution law and its parameters have been defined as well.
Key words: comparator; digital comparator; LT, EQ, GT signs; binary numbers; binary-decimal numbers; 8421, 8421^{+1}, 2421 codes; floating point numbers; IEEE-754 standard; latency; Pierson criteria; geometric distribution.

Въведение

Разглежда се операция сравнение *Compare(x,y),* на две числа *x* и *y*, която е често срещана в алгоритмите. За изпълнение на тази операция в цифровите процесори се включват съответните машинни команди [1], [2], [6]. Операнди *x* и *y* на тези команди са числа със знак и могат да бъдат представени в различни форми. Установяването на отношението между двете числа

$$x * y, \qquad * \in \{<, =, >\} \tag{1}$$

цели да формира и присвои съответните истинни стойности на признаците *LT (Less Than), EQ (equal)* и *GT (Greater Than)* според следните правила:

$$if \ \ (x < y) \ \ then \ \ LT = 1, \ EQ = 0, \ GT = 0. \tag{2}$$

$$if \ \ (x = y) \ \ then \ \ LT = 0, \ EQ = 1, \ GT = 0. \tag{3}$$

$$if \ \ (x > y) \ \ then \ \ LT = 0, \ EQ = 0, \ GT = 1. \tag{4}$$

Операцията сравнение се изпълнява в аритметично-логическото устройство на процесора чрез привеждане на отношението (1) към еквивалентното му с привеждане на операндите в инверсен машинен код

$$x \ * \ y \quad \rightarrow \quad (x - y) \ * \ 0 \ . \tag{5}$$

Като се отчете още, че числата могат да бъдат представени в различни бройни системи, в различни машинни кодове, а така също и в различни форми и формати, с голяма увереност може да се твърди, че такова изпълнение на операция сравнение е възможно най-бавното.

В същото време съществуват хардуерни структури, които съдържат в себе си като логически възли комбинационни компаратори, както и съществуват интегрални схеми, съдържащи единствено цифрови компаратори, позволяващи каскадно удължаване на разрядността на числата [3], [4], [5]. В тези случаи комбинационният компаратор е синтезиран въз основа на алгоритъма за без знаково сравнение на две двоични комбинации, при което основното му достойнство е високото бързодействие.

Настоящото изследване е посветено на един принципно нов подход за изграждане на многофункционални цифрови компаратори, способни на висикоскоростно изпълнение на операция сравнение. Многофункционалността се изразява в заложената мултиформеност и мултиформатност, позволяваща сравнението на:

1. Цели и дробни двоични числа, представени в обратен и допълнителен кодове, както и в техните модификации;
2. Цели и дробни двоично-десетични числа, кодирани в код 8421, 8421^{+3}, 4221, 2421, представени в обратен и допълнителен кодове, както и в техните модификации;
3. Числа представени във форма с плаваща запетая според стандарта IEEE-754.

Възможностите на такава логическа схема могат да я определят като универсална или още като инвариантна към фóрмата и формата за представяне на сравняваните числа.

Основни съображения относно представянето на числата

Какъвто и да е видът на представените числа, всеки от горните случаи се разпада на 4 отделни подслучая в зависимост от знаците им. Практически това е единственото общо нещо между изброените 3 случая. Налага се изводът, че независимо от всички останали характеристики, основавайки се на алгебрическия смисъл на числата, сравнението на знаците им еднозначно определя стойностите на признаците (2), (3) и (4), когато те са различни както следва

$$\begin{aligned} &if \ \ ((x > 0) \cap (y < 0)) \ \ then \ \ LT = 0, \ EQ = 0, \ GT = 1 \, ; \\ &if \ \ ((x < 0) \cap (y > 0)) \ \ then \ \ LT = 1, \ EQ = 0, \ GT = 0. \end{aligned} \tag{6}$$

Техническата реализация на сравнението на знаковите битове е облекчено от факта, че във всички форми, в които се представят числата, знаковият бит е най-левия в разрядната мрежа.

В случаите, когато знаците на числата са еднакви, признаците (2), (3) и (4) следва да бъдат определени след сравнение на останалите битове на комбинациите, представящи числата. Това налага разглеждането на случаите по отделно.

А) Двоични числа

Най-често двоичните числа се представят и интерпретират като цели числа. Тъй като общият случай предполага числа със знак, то за машинно представяне се използва обратен

или допълнителен код, подпомагащи изпълнение на операция изваждане. За n-битова разрядна мрежа с дясно фиксирана запетая [6] тези кодове се дефинират така

$$[x]_{OK}=\begin{cases} x, & if\ \ x\ge 0\,; \\ 2^n-1-|x|, & if\ \ x\le 0\,. \end{cases} \qquad [x]_{ДК}=\begin{cases} x, & if\ \ x\ge 0\,; \\ 2^n-|x|, & if\ \ x<0\,. \end{cases} \tag{7}$$

където чрез n е означена дължината на мултиформатната разрядна мрежа. Различните формати са възможни благодарение на правилото за знаково разширение на числата, според което старшите незначещи цифри в разрядната мрежа, при нейното удължаване, съвпадат със знаковата цифра. При по-късите формати се изключва възможността за препълване. Последното може да бъде откривано хардуерно. Аналогична на (7) зависимост дефинира инверсните кодове и за числата с ляво фиксирана запетая.

В първия случай на (7), когато и двете числа са положителни, отношението на двата допълнителни кода $[x]_{ДК} * [y]_{ДК}$ (или $[x]_{ОК} * [y]_{ОК}$) е еквивалентно на отношение (1) и може да се изчисли чрез алгоритъма на беззнаковото сравнение.

Във втория случай на (7), когато и двете цели числа са отрицателни, отношението на двата инверсни кода се изразява съответно чрез отношението на числата

$$\begin{gathered} (2^n-1-|x|)\ *\ (2^n-1-|y|)\ ; \\ (2^n-|x|)\ *\ (2^n-|y|) \qquad . \end{gathered} \tag{8}$$

което очевидно също е еквивалентно на (1).

В резултат може да бъде обобщено, че алгоритъмът на без знаковото сравнение може да се използва за сравняване както на числа без знак, така и на числа със знак, представени в инверсни машинни кодове, включително и модифицирани такива, които имат по 2 знакови бита. Това важи и за числа с ляво фиксирана запетая, както и за числа, структурирани с цяла и дробна част. С други думи положението на запетаята е без значение, тъй като тук се имат предвид само числа, представени в позиционни бройни системи.

Когато сравняваните числа са представени в обратен код, се налага предварително автоматично разпознаване и игнориране на знаковите нули (+0, -0), които имат място в този код. Това лесно може да бъде осигурено с хардуерни средства.

Казаното в този раздел е илюстрирано със следния числен пример

x=-16, y=-19, n=8 [b], $[x]_{ДК}=1\ 1110000$, $[y]_{ДК}=1\ 1101101$, GT=1.

$$\begin{array}{l} 1\ \ 1110000 \ \ \rightarrow x \\ \underline{1\ \ 1101101 \ \ \rightarrow y\ \ *} \\ \qquad\quad GT=1 \quad (x>y) \end{array}$$

Б) Двоично-десетични числа

Десетичните числа се представят като наредена последователност от кодовите комбинации на своите десетични цифри. Известни са различни кодове, като например (8421), (8421^{+3}), (2421) и др. Най-широко се прилага първият от посочените. За числата със знак се използват специални схеми [6], [10] за получаване на техния инверсен машинен код. Последният се дефинира аналогично на (7) така

$$[x]_{OK}=\begin{cases} x, & if\ \ x\ge 0\,; \\ q^n-1-|x|, & if\ \ x\le 0\,. \end{cases} \qquad [x]_{ДК}=\begin{cases} x, & if\ \ x\ge 0\,; \\ q^n-|x|, & if\ \ x<0\,. \end{cases}, \tag{9}$$

където с q е означена основата на бройната система, в която е представено числото, като в частност тук се има предвид q=10. С n отново е изразена дължината на мултиформатната разрядна мрежа. За всеки 2/10-чен код е известна [6] алгоритмична схема за постигане на зависимостта (9).

Прилагането на алгоритъма на без знаково сравнение за проверка на отношението $[x]_{ДК} * [y]_{ДК}$ (или $[x]_{ОК} * [y]_{ОК}$) на 2/10-чни числа напълно съответства на зависимостите (2), (3), (4) и (6). При изпълнение на този циклически алгоритъм на сравнение се подлагат две цифри в текущия i-ти разряд $dx_i * dy_i$. Тъй като тези цифри са представени с двоични комбинации, това отношение се проверява бит по бит по силата на алгоритъма на двоичното без знаково сравнение. От това следва, че това отношение ще бъде вярно, ако за 2/10-чния код е в сила свойството монотонност [6]. Това свойство е в сила и за кодовите комбинации на десетичните цифри в инверсни машинни кодове на числата със знак. От тук следва окончателният извод, че алгоритъма на двоичното без знаково сравнение може да се приложи и върху различно кодирани 2/10-чни числа. Алгоритъмът е инвариантен към положението на запетаята на числата.

Казаното в този раздел е илюстрирано със следния числен пример

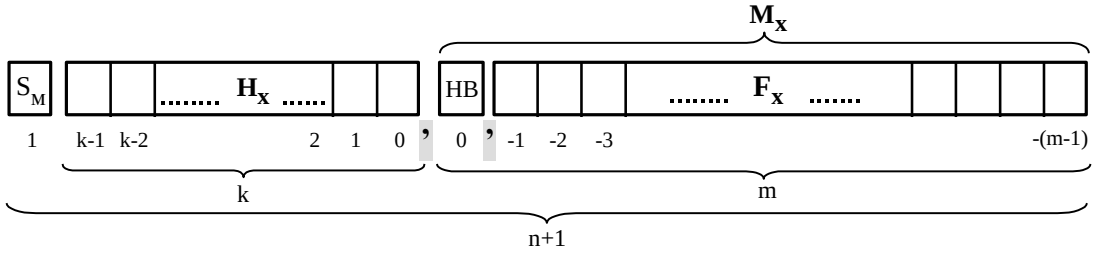
BCD:8421; x=-6170, y=-3369, n=5 [pos.], LT=1.

$$[x]_{ДК} = 1 \quad 1001 \quad 0011 \quad 1000 \quad 0011 \quad 0000,$$
$$[y]_{ДК} = 1 \quad 1001 \quad 0110 \quad 0110 \quad 0011 \quad 0001,$$

$$\begin{array}{llllll} 1 & 1001 & 0011 & 1000 & 0011 & 0000 \quad \rightarrow x \\ 1 & 1001 & 0110 & 0110 & 0011 & 0001 \quad \rightarrow y^{*} \\ \hline & & LT{=}1 & & & (x < y) \end{array}$$

В) Числа във форма с плаваща запетая

Разглежда се структура на разрядната мрежа според стандарта IEEE 754 с техника на скрития бит [6], [8], имаща вида



Фиг. 1. Структура на разрядната мрежа

в която означенията са:

$S_М$ – знаков бит на мантисата (на числото);

H_X – характеристика. За характеристиката са в сила следните зависимости:

$$H_x = p_x + (D-1) = (p_x - 1) + D, \qquad D = 2^k; \tag{10}$$

HB – скрит бит;

M_X – нормализирана мантиса. Има вида: $M_X = 1,F_X$;

F_X – фракция.

Мултиформатността се осигурява от променящите се дължини k (8, 11 или 15 бита) и m (23, 52 или 64 бита), дефинирани от стандарта.

Мантисата е число със знак и се представя в прав код. Това означава, че отношението (1) може да бъде определено чрез алгоритъма на без знаковото сравняване. Това отношение обаче следва да бъде съобразено с характеристиките на числата. Характеристиките се възприемат като числа без знак и по стойност също могат да се сравнят чрез алгоритъма на без знаковото сравняване. Тяхното отношение обаче трябва да се приложи върху стойностите на порядъците, които са числа със знак.

Според (10) отношението $H_x * H_y$ се разглежда във вида

$$(p_x - 1) + 2^k \ * \ (p_y - 1) + 2^k \ , \tag{11}$$

който може да бъде сведен до

$$p_x \ * \ p_y \ . \tag{12}$$

Последното отношение обаче не винаги е еквивалентно на отношението на характеристиките, тъй като порядъците са числа със знак, а те могат да се случват в 4 различни комбинации.

Комбинациите от различни знаци съответстват на правилно определено чрез сравняване на характеристиките отношение, тъй като отрицателният порядък води до операция изваждане (виж (10)), което силно отдалечава по стойност съответната характеристика от другата. В случай, че знаците са положителни, отношението на характеристиките е в пълно съответствие с отношение (12). В последния случай, когато знаците на порядъците са отрицателни, в резултат на операция изваждане, отношение (12) също в съответствие с отношение (11). В случай на две отрицателни числа, по-голямо е числото с по-малкия порядък.

Казаното в този раздел е илюстрирано със следните числени примери за 4-те знакови комбинации на числата в отношението $x * y$:

Пример 1. $x = +5.10^{-5}, \ y = -6.10^{+2}$.

В този случай се получава признак *GT*=1, тъй като *x*>0, а *y*<0 ;

Пример 2. $x = -5.10^{-5}, \ y = +6.10^{+2}$.

В този случай се получава признак *LT*=1, тъй като *x*<0, а *y*>0 ;

Пример 3. $x = +5.10^{-5}, \ y = +6.10^{+2}$.

В този случай знаците на числата са еднакви и сравнението се пренася върху старшите битове на характеристиките. Тъй като $p_x < 0$, старшият бит на характеристиката H_x ще съдържа нула, т.е. $H_x[k-1] = 0$. В същото време $p_y > 0$. Това означава, че старшият бит на характеристиката H_y ще съдържа единица, т.е. $H_y[k-1] = 1$. В резултат от сравнението на старшите битове на характеристиките за отношението на числата ще бъде определен признак *LT*=1, т.е. 0,00005 < 600.

Пример 4. $x = -5.10^{-5}, \ y = -6.10^{+2}$.

В този случай знаците на числата са отново еднакви и характеристиките са същите. Сравнявайки ги отново ще се получи същия признак *LT*=1, което за характеристиките е вярно, но за числата като цяло не е вярно, тъй като отчитайки знаците -0,00005 > -600

Анализът и приведените числени примери за сравнение на числа във форма с плаваща запетая изявяват два случая:

- Резултатът от без знаковото сравнение е верен;
- Резултатът от без знаковото сравнение е инверсен. Явява се при 2 отрицателни числа.

Изключението, което се явява в работата на компаратора във вторият случай, откроен по-горе, неизбежно изисква предварителна инициализация на логическата схема. Тя може да се постигне чрез допълнителен сигнал FPN (*Floating-Point Numbers*), определящ числата като представени във форма с плаваща запетая.

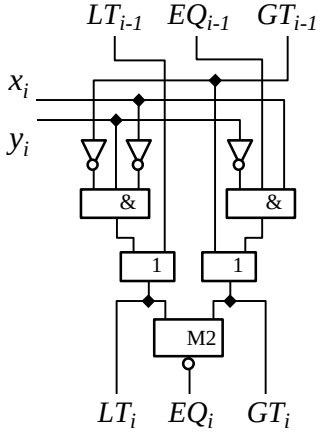
Логическа схема на компаратора

Логическата схема на компаратора е синтезирана в съответствие с алгоритъма за без знаковото сравнение и представлява последователност от еднобитови компаратори. Принципната логическа схема за всички средни *i*-ти разряди $i = \overline{(n-2),1}$ е представена на фигура 2. Тя е синтезирана според методиката, представена в [5], и има следната логика

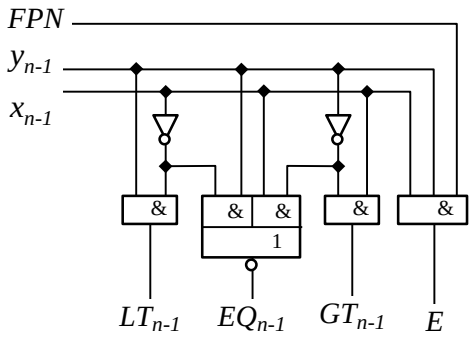
$$LT_i = LT_{i+1} \cup (\overline{GT_{i+1}} \cap \overline{x_i} \cap y_i);\ \ GT_i = GT_{i+1} \cup (EQ_{i+1} \cap x_i \cap \overline{y_i});\ \ EQ_i = \overline{LT_i \oplus GT_i}\ . \quad (13)$$

Логическата схема на компаратора в най-старшия (*n-1*)-ви разряд (фигура 3) е синтезира като функция и от инициализиращия сигнал *FPN*. Така, когато числата са във форма с плаваща запетая, в този разряд се генерира разрешението *E* (*Enable*), което се използва в най-младшия (нулевия) разряд на компаратора. Логиката на схемата е следната

$$LT_{n-1} = \overline{x_{n-1}} \cap y_{n-1};\ GT_{n-1} = x_{n-1} \cap \overline{y_{n-1}};\ EQ_{n-1} = \overline{x_i \oplus y_i};\ E_i = FPN \cap x_i \cap y_i\ . \quad (14)$$



Фиг. 2. Логическа схема на *i*-тия бит

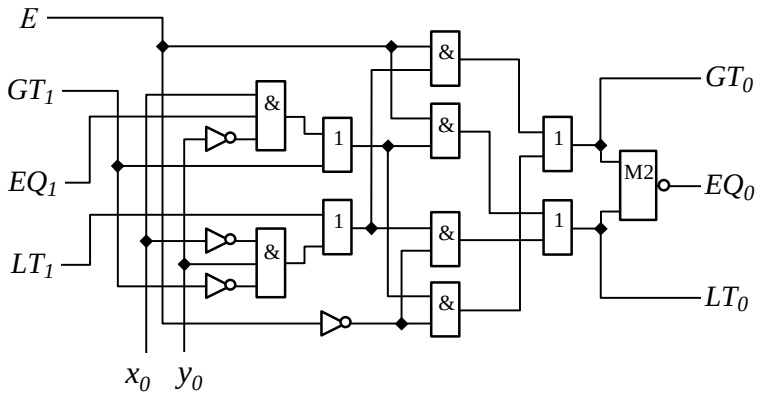


Фиг. 3. Логическа схема на старшия бит

Логическата схема на компаратора в най-младшия *0*-лев разряд е представена на фигура 4. Както се вижда, изходните стойности на признаците са функция от разрешението *E*, чиято логика е следната

$$\begin{aligned} LT_0 &= \overline{E} \cap \left(LT_1 \ \cup\ (\overline{GT_1} \cap \overline{x_0} \cap y_0)\right) \ \cup\ E \cap \left(GT_1 \ \cup\ (EQ_1 \cap x_0 \cap \overline{y_0})\right); \\ GT_0 &= \overline{E} \cap \left(GT_1 \ \cup\ (EQ_1 \cap x_0 \cap \overline{y_0})\right) \ \cup\ E \cap \left(LT_1 \ \cup\ (\overline{GT_1} \cap \overline{x_0} \cap y_0)\right); \\ EQ_0 &= \overline{LT_0 \oplus GT_0}\ \ . \end{aligned} \quad (15)$$

Разрешението влияе на окончателните резултати, само в случай че числата са във форма с плаваща запетая и са отрицателни и не са напълно еднакви.



Фиг. 4. Логическа схема на младшия бит

Статистически подход за идентификация на закона на разпределение на латентността на компаратора

Времето за превключване на логическата схема на компаратора е основен технически параметър, който го характеризира. В случая, времето за превключване е пропорционално на дължината на верижката, образувана от срещнатите в посока към младшите разряди последователно еднаквите старши цифри в двете числа. При среща на различни цифри в текущия бит, признаците се определят еднозначно и останалите до най-младшия бит цифри, не се нуждаят от сравнение. Така, разглеждана като случайна величина, времето за превключване на логическата схема, т.е. на латентността й, се нуждае от специално изследване. За целта беше организиран и проведен числен експеримент чрез програмен модел на компаратора, създаден на програмния език C++. В модела на сравнение са подлагани двойки числа, генерирани като псевдослучайни с помощта на функция *boost::random* от библиотека "Boost" [9]. Сравняваните числа са генерирани в 32-битовия диапазон на разрядната мрежа, като получените дължини за латентността на компаратора при всяко сравнение, са натрупани в извадки с обем 1000 и 5000 сравнения.

Статистическото изследване на получените извадки има за цел да намери закона за разпределение на случайната величина, както и неговите параметри. Последните са необходими за оценка на производителността на компаратора, когато е използван в други изчислителни структури.

Като модел на верижката от фактически извършени сравнения с резултат равно (*EQ*) се използва величината (*w-1*). Всеки бит, в който очакването да бъде прекратена верижката от последователни съвпадения не се реализира, се оценява с вероятност (*1-p*). Тогава вероятността, верижката да не бъде прекъсната след (*w-1*)-тия бит е $(1\text{-}p)^{(w-1)}$. С отчитане на условието, че в бит *w* събитието прекъсване ще се състои с вероятност [*1*-(*1-p*)]=*p*, то законът за разпределение има вида

$$f = p.(1-p)^{(w-1)} \; ; \quad w = \overline{1,n}, \;\; p \geq 0 \, . \tag{16}$$

Видът на тази функция дава основание да бъде издигната нулева хипотеза H_0 за статистическата апроксимация на латентността на компаратора със закона за геометрично разпределение. За определяне на параметъра *p* в (16) и неговата адекватност към експерименталните данни е използван критерия на Пирсон [11]

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - f_i.N)^2}{f_i.N} \; , \tag{17}$$

където: i - е номер на интервала; N - обем на статистическата извадка;

n_i - брой попадения в i-тия интервал. Случайната величина w има 32 възможни стойности, които могат да се разпределят в 32 интервала на 32 битовата разрядна мрежа;

f_i - честота на попадения по теоретичния закон.

Таблица 1. За извадка с обем 1000 сравнения

w	n_i	$p_i=n_i/1000$	f_i	fn_i
1	473	0,473	0,493	493
2	260	0,260	0,25	250
3	142	0,142	0,1267	127
4	75	0,075	0,0642	64
5	28	0,028	0,0326	33
6	22	0,022	0,0165	17

Числените резултати от проведеното изследване, получени в средата MATLAB, са представени в таблици 1 и 2, където са използвани следните означения:

w - номер на интервала, съответстващ на номера на изходящия бит на верижката;

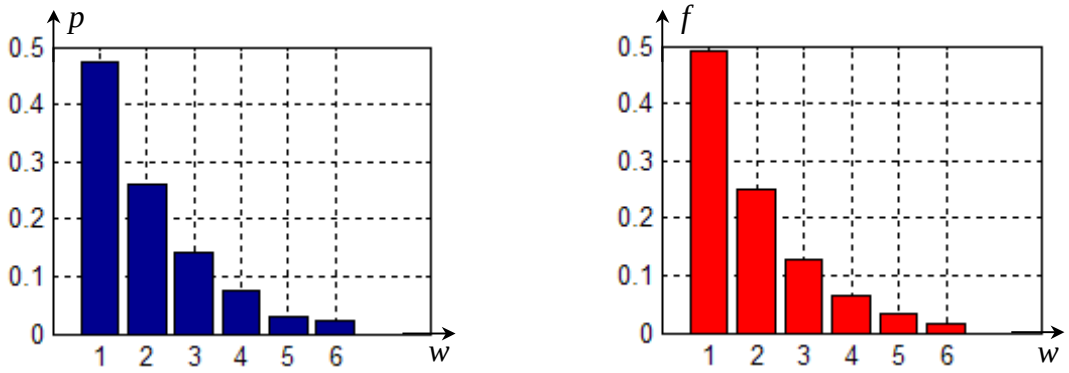
p_i - относителна честота на попаденията в i-тия интервал;

fn_i - брой попадения, определени по теоретичния закон.

За всяка извадка са определени максималната и минимална стойност на случайната величина, математическото очакване M, както и нейната дисперсия S^2.

В първата извадка максималната дължина на латентността е 13 бита, а минималната 1 бит. Математическото очакване има стойност M=2,035, а дисперсията – S^2=2,0658. Вероятността, събитието да се състои, е p=0,493. Числената стойност на критерия $\chi^2 = 7{,}1018$ е по-малка от теоретичната $\chi_T{}^2(0{,}05;5) = 11{,}07$, определена при ниво на значимост v=0,05 и степени на свобода k=max(w)-1=5, откъдето следва, че представените данни не противоречат на хипотезата за геометрично разпределение.

На фигура 5 са представени хистограмите на честотата на попадение p_i (вляво) и теоретичното разпределение f (вдясно).



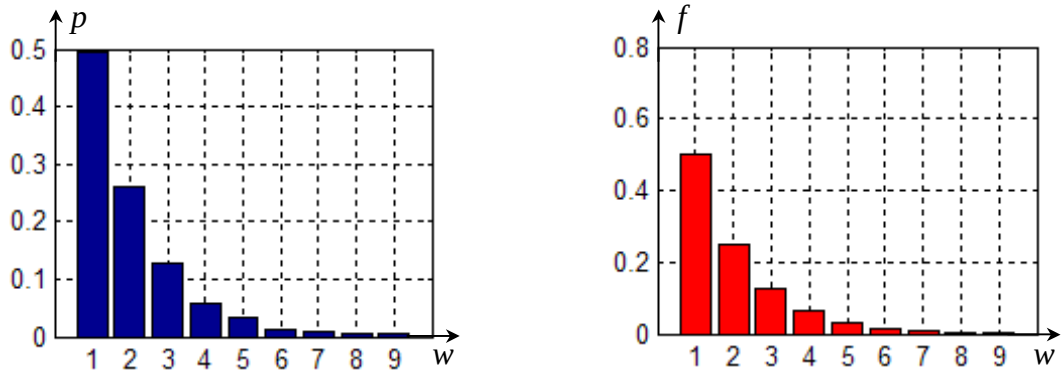
Фиг. 5. Разпределение на латентността в извадка с обем 1000 сравнения

Във втората извадка максималната дължина на латентността е 17 бита, а минималната 1 бит. Математическото очакване има стойност M=1,9998, а дисперсията – S^2=1,9918. Вероятността, събитието да се състои, е p=0,5004. Числената стойност на критерия $\chi^2 = 7{,}146$ е по-малка от теоретичната $\chi_T{}^2(0{,}05;8) = 15{,}507$, определена при ниво на значимост v=0,05 и степени на свобода k=max(w)-1=8, откъдето следва, че представените данни не противоречат на хипотезата за геометрично разпределение.

Таблица 2. За извадка с обем 5000 сравнения

w	n_i	$p_i=n_i/1000$	f_i	fn_i
1	2467	0,4934	0,5004	2502
2	1296	0,2592	0,25	1250
3	635	0,127	0,1249	624
4	290	0,058	0,0624	312
5	162	0,0324	0,0312	156
6	64	0,0128	0,0156	78
7	47	0,0094	0,0078	39
8	20	0,004	0,0039	19
9	19	0,0038	0,0019	10

На фигура 6 са представени хистограмите на честотата на попадение p_i (вляво) и теоретичното разпределение f (вдясно).



Фиг. 6. Разпределение на латентността в извадка с обем 5000 сравнения

Заключение

Синтезът на цифров компаратор с множеството различни характеристики се оказа възможен. Синтезираната логическа схема прилага последователно сравнение, но може да бъде преструктурирана чрез подходи като описания в [7].

Схемата е мултиформена, защото е инвариантна към запетаята на числа, представени в позиционни бройни системи. Тя е инвариантна още към инверсните машинни кодове (обратен, допълнителен и техните модификации).

Схемата е мултиформатна, тъй като е инвариантна към дължината на разрядната мрежа, в която са представени числата, което се дължи на свойствата на инверсните машинни кодове.

Схемата е инвариантна още към кода за представяне на десетичните цифри, стига той да притежава свойството монотонност, което е показано тук за няколко от най-прилаганите.

Схемата проявява своите универсални възможности и върху числа представени във форма с плаваща запетая според стандарта IEEE-754. Доказано е, че синтезираната схема успешно се справя в три от четирите случая на комбиниране на знаците на такива числа. Изключението, което може да се яви при числа с отрицателни знаци, успешно се дешифрира, при това с минимални апаратни средства, не влияещи на латентността ѝ. Мултиформатността на схемата е в сила и върху числата с плаваща запетая благодарение на адитивната функционалност, според която е дефинирана характеристиката на числата.

Времето за превключване на цифровия компаратор е функция от самите числа, ето защо то е променлива величина, която има случаен характер и геометрично разпределение. Статистическите резултати налагат обобщението, че в рамките на допустимото ниво на значимост за критерия на Пирсон, очакваното време за превключване на компаратора е в границите на 1/3 от неговата дължина.

Литература

[1]. Hennessy J.L., D. A. Patterson, Computer Architecture. A Quantitative Approach. 4-ed., ISBN: 1-55860-724-2, Morgan Kaufman Publishers, Amsterdam, 2007.
[2]. Tanenbaum A. S., Structured Computer Organization. 5-ed, Prentice Hall, ISBN 0-13-148521-0, New Jersey, 2006.
[3]. Wakerly J. F., Digital Design. Principles & Practices. 3-ed, Prentice Hall, ISBN 0-13-769191-2, New Jersey, 2000.
[4]. Lala P. K., Principles of modern digital design. John Wiley & Sons, ISBN 978-0-470-07296-7, New Jersey, 2007.
[5]. Enoch O. Hwang, Digital Logic and Microprocessor Design with VHDL. La Sierra University, Riverside, ISBN: 0-534-46593-5, 2005.
[6]. Тянев Д. С., Организация на компютъра. Том 1, ISBN: 978-954-20-0412-7. Технически Университет - Варна, 2008.
[7]. Stine J. E., M. J. Schulte, A Combined Two's Complement and Floating-Point Comparator, IEEE International Symposium on Circuits and Systems, 2005. Kobe. ISBN: 0-7803-8834-8, vol. 1, pp. 89-92.
[8] Standards Committee of the IEEE Computer Society, IEEE Standard 754 for Binary Floating Point Arithmetic. IEEE Press, August 1985.
[9]. C++ source libraries: http://www.boost.org/ .
[10]. BCD 2421-code: http://www.tyanev.com/home.php?lang=bg&mid=18&mod=1&b=7&s=403
[11]. Генов Д. Г., Моделиране и оптимизация на производствени процеси, ISBN: 954-20-0143-6. Технически Университет - Варна, 2000.

За контакти:
Димитър С. Тянев,
dstyanev@yahoo.com

Димитър Г. Генов
dggenov@yahoo.com

Веселин В. Стефанов
veski4a92@gmail.com

Мартин Д. Георгиев
dimitrov.martin1@gmail.com

Стефка И. Попова
s_ivanova@abv.bg

АСИНХРОННИ ЦИКЛИЧЕСКИ МИКРОКОНВЕЙЕРНИ СТРУКТУРИ С МНОГОТАКТОВИ ТЕЛА

Димитър С. Тянев

Резюме: Представен е нов метод за хардуерна реализация на циклически алгоритмични структури с асинхронна микроконвейерна организация. Изследването е върху цикли със следусловие и многотактови циклически тела с произволна структура. За цикли с отнапред неизвестен брой повторения представената конвейерна организация на циклическото тяло е единствено възможната.. Формулирани са и са решени четири нови за конвейерната организация задачи. Представен е синтезът и принципни логически схеми на необходимите конвейерни автомати.

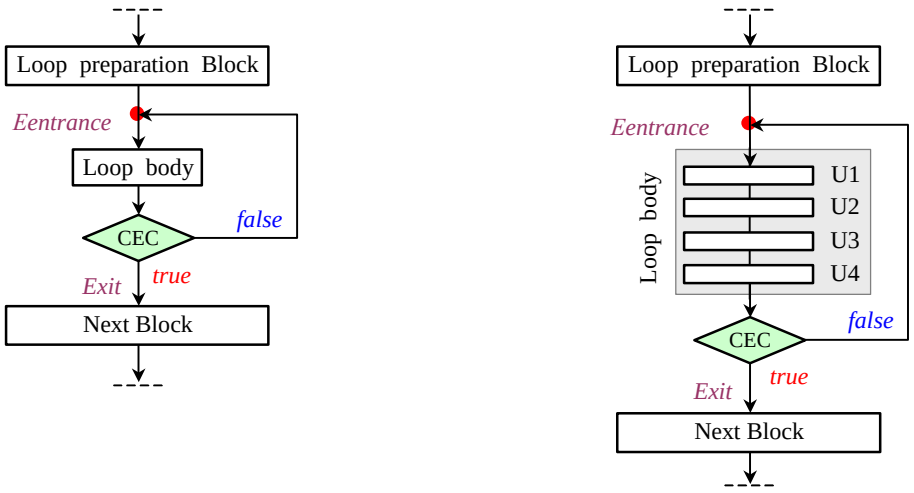
Asynchronous micro-pipeline loop-structures with multi-stage bodies

Dimitar S. Tyanev

Abstract: A new method for hardware realization of algorithmic loop-structures with asynchronous micro-pipeline organization is representative. The study is on loops with post-condition and on multi-stage bodies with common structures. For loops with beforehand unknown number of repetitions, the presented here pipeline organization is the only possible. Four new tasks for pipeline organization are formulated and solved. The synthesis and logical circuit of the necessary pipeline controllers are presented.

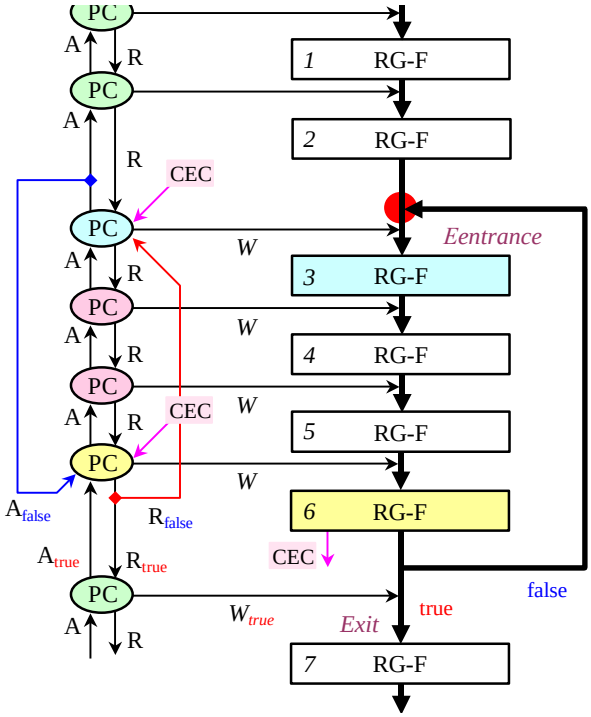
1. Постановка на задачата

Разглежда се микроконвейерна реализация на циклически алгоритмични структури, които притежават линейно многотактово тяло. На фиг.1 е илюстрирано разбирането за тази структура. Цикъл с еднотактово микроконвейерно звено в тялото е илюстриран чрез лявата блок-схема. Многотактовият вариант на тялото на цикъла е показан чрез дясната блок-схема, като 4-степенна линейна последователност от звената *U1*, *U2*, *U3* и *U4*. Принципна разлика между двата варианта няма. Целта на изследването е разработване на метод за управление на такива циклически структури с асинхронна микроконвейерна организация. Първоначалната представа за поставената цел е илюстрирана на фиг.2. Както може да се види, микроконвейерните звена (представени чрез своите регистри фиксатори RG-F) се управляват от конвейерни автомати *PC* (*Pipeline Controller*), които генерират импулсите за запис на данни след успешно "ръкостискане" със своите съседи.



Фиг. 1. Алгоритмична структура цикъл със следусловие

Ръкостискането се основава на сигналите заявка *Request* (*R*) и потвърждение *Acknowledgement* (*A*), които автоматите използват в диалога.



Фиг. 2. Асинхронна конвейерна организация на цикъл

Изобразените звена са номерирани от 1 до 7. Тези номера ще бъдат използвани и за означаване на конвейерните автомати, на техните сигнали или връзки, но няма да имат нищо общо с поредовия номер на зарежданите в конвейера задачи.

2. Хардуерен аспект на анализа

В циклическата структура от фиг.2 има две съществени точки. Това са входната точка (*Entrance*) и изходната точка (*Exit*). По своята природа, тези две точки са свързани чрез обратната връзка, която съответства на алгоритмичния преход "лъжа" на условието за край на цикъла *CEC* (*Condition of End Cycle*). Управлението на микроконвейерните звена в тези две точки е в зависимост от логическата стойност на условието за край *CEC*. Конвейерните автомати на входното и на изходното звена остават свързани през цялото време на циклическото изчисление. Заявката, която генерира автоматът в изходната точка, трябва да се разклонява, пренасочвайки актуалната си стойност в зависимост от условието *CEC* или към автомата на входното звено (R_{false}), или към автомата на звеното в изхода на цикъла (R_{true}). В същото време този автомат ще трябва да формира своето потвърждение въз основа на две входни за него потвърждения – това, което идва от автомата на входното звено (A_{false}) и това, което идва от автомата на звеното по изхода на цикъла (A_{true}).

Що се отнася до конвейерния автомат, управляващ входното звено на цикъла, той е свързан с две входни заявки – тази, която идва от звеното, предхождащо входната точка (*R*), и идващата от автомата на изходното звено (R_{false}). В отговор трябва да връща потвърждение или към предходния автомат (*A*) или към автомата в изходната точка (A_{false}).

Началният анализ на логическата структура от фиг.2 изявява две задачи. Задачите са за синтез на принципните логически схеми на конвейерните автомати в двете характерни точки, тъй като те имат значително по-сложна система за диалог в сравнение с останалите автомати.

3. Организационен аспект на анализа

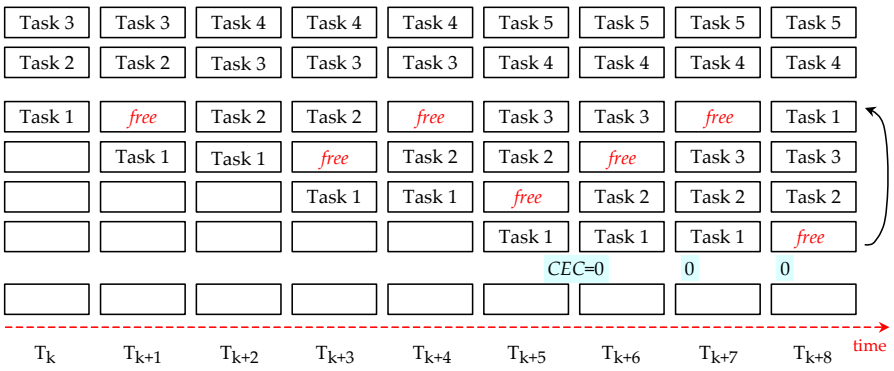
Формулираните по-горе задачи се отнасят само до чисто хардуерния аспект на цялостния проблем, при това в един начален и неокончателен за него вариант. Разглежданата цикличес-ка структурата е интересна още в организационен аспект. Нейното функциониране следва да бъде организирано конвейерно. Разглеждана обобщено като "едрозърнеста", структурата се определя като многотактова, тъй като е обхваната от обратна връзка. В същото време обаче като "дребнозърнеста", нейното циклическо тяло е многотактово, представено в примера чрез линеен конвейерен участък с 4 степени. При тези условия, функционирането на циклическата структура може да се организира по два различни начина.

Първият начин съответства на функционирането на еднотактовата циклическа структура (фиг.1), за която са познати различни варианти [1]. Става дума за функциониране, при което, когато една задача постъпи в структурата, тя не я освобождава, докато изчисленията не завършат. Дори когато циклическото тяло е многотактово, както бе определено, задачата се превърта многократно, като преминава последователно от звено към звено в тялото. Този вид организация се определя като ниско производителна, защото във всеки момент в конвейерното тяло работи само едно звено, а останалите се намират в изходно състояние. Този вариант на организация е практически безинтересен и ще бъде използван единствено като основа за сравнение.

По-високата производителност изисква ангажирането на всички свободни звена в циклическото тяло на структурата. Конвейерната организация позволява в тялото на цикъла да се въртят едновременно повече от една задачи. Така например, след навлизане на задача №1 в тялото на цикъла, при което звеното във входната точка се освобождава, а входната даннова шина все още не е изключена, в цикъла може да постъпи задача №2. Аналогично, когато тя, следвайки задача 1, освободи звеното във входната точка, на нейно място може да постъпи задача 3.

Като има предвид, че в асинхронните конвейери регистрите фиксатори се изграждат от тригери със структура *Latch*, едно циклическо тяло с m на брой степени може да бъде заредено безпроблемно максимум с $(m-1)$ на брой задачи. Това е така, защото m-степенното кръгово изместване на задачите от звено към звено в тялото на цикъла не може да се осъществи без наличие на свободно звено. Обратната връзка на цикъла не би могла да върне във входното звено задача, идваща от изходното звено, ако входното не е свободно.

Процесът на зареждане и превъртане на задачи в примерната структура е илюстриран на фиг.3. Първото връщане на задача 1 по обратната връзка във входното звено е изобразено в такт T_{k+8}. Преходът на задача 1 от звено 6 в звено 3 се дължи на нулевата стойност на условието *CEC*, която звено 6 изчислява за нея. Тази логическа стойност определя данновите връзки в изходната и входната точки, както беше пояснено в началото.

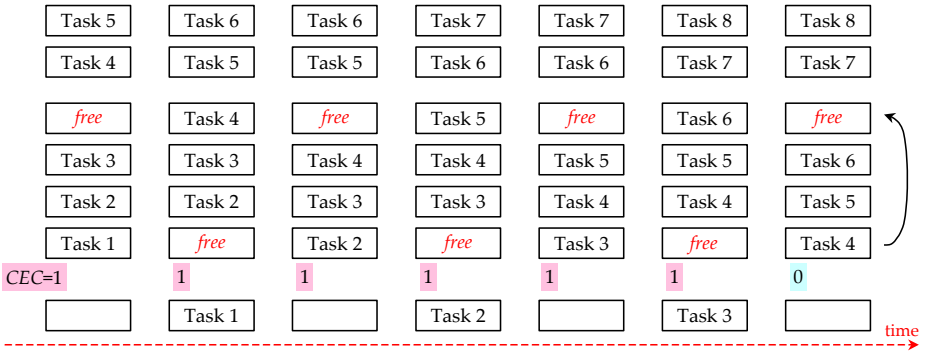


Фиг. 3. Навлизане на много задачи в цикъла и превъртане

Казаното до момента все още не изчерпва изясняването на организационния аспект на функционирането на конвейеризираната циклическа структура. За целите на по-нататъшния анализ ще въведем допълнителна конкретност.

4. Цикъл с предварително известен брой повторения

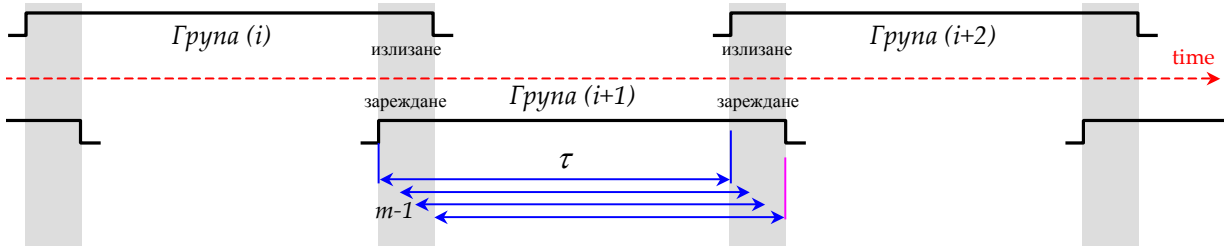
Когато реализираният цикъл е от вида с предварително известен брой повторения, заредените в неговото тяло задачи се превъртват еднакъв брой пъти. От това следва, че тяхната последователност ще се запази и до момента на последното повторение. Условието за край на цикъла *CEC* ще се изпълни за първи път най-напред за задача 1 и тогава тя трябва да излезе от цикъла. След нея по същата причина последователно ще напуснат цикъла и останалите задачи от групата. След като звеното във входната точка бъде освободено, тъй като обратната връзка по това време ще бъде изключена (*CEC*=1), в него може да постъпи нова задача (втората колонка в рисунката от фиг.4). Така процесът на последователно излизане от цикъла на трите изпълнени задачи №(1,2,3) ще бъде съпроводен с последователно зареждане на 3 нови задачи №(4,5,6). Фигура 4 илюстрира тези процеси.



Фиг. 4. Процес на излизане на задачи от цикъла

Вижда се, че в последния изобразен такт за задача 4 се изчислява условието *CEC*=0. С това започва превъртането в тялото на цикъла на втората група задачи.

На кратко може да се каже, че ако тялото на цикъла е конвейерно реализирано чрез *m* на брой степени, структурата от фигура 2 може да работи по едновременното изпълнение на (*m-1*) на брой задачи. В процеса на излизане от цикъла на завършилите задачи, в него се зареждат също толкова нови задачи. Ако се приеме, че броят на циклическите повторения е *n* на брой, то латентността на цяла група от (*m-1*) задачи е почти колкото за една единствена задача. Процесът на излизане на задачи от цикъла, заедно с процеса на зареждане, може да бъде представен чрез времедиаграмата от фигура 5.



Фиг. 5. *Циклическа активност на 3 групи от по (m-1) задачи*

На времедиаграмата активните изчисления по задачите от дадена група се застъпват във времето, когато те излизат от цикъла, с навлизането на следващата група задачи. Латентността τ на всяка задача се представя със следния израз

$$\tau = n.\sum_{k=1}^{m} t_k \quad , \tag{1}$$

където с t_k е означена латентността на k-тото звено.

Ако се приеме (в най-лошия случай), че латентността t_k на всяко звено е константа, равна на максимално възможната за съответното звено, следва че и оценката (1) е константа. Последното дава правото да се твърди, че при тази организация, конвейерната циклическа структура е най-малко (*m-1*) пъти по-производителна по сравнение с еднозадачната организация.

5. Място и време на събитията

За да бъдат изявени настъпващите събития, ще бъде анализиран по-подробно процеса на преминаване на задачите през изходната и входната точки. Трансферът на данните в тези точки зависи от стойността на условието *CEC*. Следва да се помни, че условието *CEC* в текущия такт се отнася само за задачата, достигнала изходната точка, и че в следващия такт в същата точка ще бъде изчислено условието на следващата задача. Анализът на функционирането на циклическата структура откроява три различни ситуации във времето:

1. Когато в тялото на цикъла навлизат новите задачи;
2. Когато тези задачи се превъртват последователно и многократно;
3. И когато задачите започват да излизат от цикъла.

След изпълнение на циклическите изчисления за няколко групи задачи, както е показано на фиг. 5, първият и третият случаи се обединяват. Между тях може да се видят отличия само при първоначално зареждане на първата група или когато при излизане на дадена група, се окаже че тя е последната.

В изходно състояние всеки конвейер се характеризира с всеобща готовност на микроконвейерните звена. Заявки няма и всички конвейерни автомати връщат един към друг сигнали, потвърждаващи готовността. В това състояние, навлизането на много задачи в тялото на цикъла, както е показано на фиг.4, изисква данновите връзки във входната му точка да са такива, че да осигурят зареждане на циклическата конвейерна структура с максимално възможния брой задачи. Докато това не се постигне, обратната връзка трябва да бъде трайно изключена. От посоченото следва, че в конвейерната структура трябва да се въведе нов елемент, чиято задача ще бъде да покаже момента, в който циклическото тяло е заредено с необходимия брой задачи. Синтезът на този елемент е следваща, трета задача. Тъй като при начално зареждане на задачи в конвейера все още не е била изчислявана стойност на условието за край на цикъла *CEC*, то управлението на данновите връзки във входната точка на цикъла по това време ще се осъществява именно от този елемент. В момента, в който задача 1 достигне звено 6 и за нея бъде изчислена стойност за условието *CEC*, управлението на данновите връзки ще поеме тази именно стойност.

По-детайлният анализ на последните два такта (k+7) и (k+8) от фиг.3 показва, че след като задача 1 в такт (k+5) попадне в звено 6, започва изчисляването на стойността на условието *CEC*. Едновременно с това останалите задачи слизат в тялото, а свободното звено *free* се премества в обратна посока. В такт (k+6) или (k+7) стойността на условието е вече изчислена и тя е *CEC*=0. Тази стойност определя данновите връзки в изходната и във входната точки, като образува обратната връзка. С това в такт (k+8) става възможен преходът на задача 1 от звено 6 в звено 3.

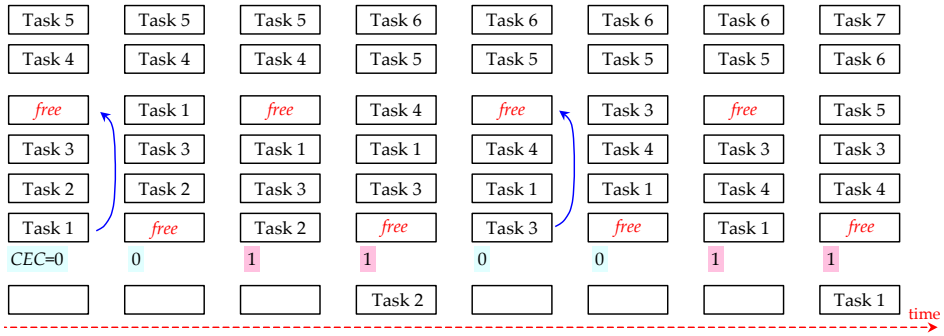
В началото на фиг.4 е изобразен обратният случай, когато *CEC*=1. При тази стойност данновите връзки в изходната и входната точки на цикъла трябва да се определят така, че обратната връзка да бъде изключена. Така задача 1 ще излезе от цикъла и ще слезе надолу в конвейера в звено 7, а в същото време в звено 3 ще бъде заредена новата задача 4. В края на

процеса, изобразен на фиг.4, в циклическата структура е заредена новата група задачи №(4,5,6). И тъй като за задача 4, която е в звено 6, е изчислена стойността *CEC*=0, обратната връзка отново е включена и тази нова група започва своето *n*-кратното превъртване в тялото на цикъла.

6. Цикъл с предварително неизвестен брой повторения

Дълготрайността на стойностите на условието за край на цикъла е характерна особеност на функционирането на разгледаната по-горе разновидност. Значително по-динамично е то когато структурата съответства на алгоритъм с неизвестен брой повторения. В този случай представата за известна статичност, че задачите ще навлязат последователно, че ще се превъртват заедно и ще излязат заедно, в реда, в който са влезли, като цялостна и постоянно задържаща се в цикъла група, губи смисъл. Конвейерното изпълнение на няколко задачи в цикъл с неизвестен брой повторения не може да запази първоначалния състав на групата от (*m-1*) на брой задачи. Тъй като броят на повторенията за всяка задача ще бъде различен, групата ще има променлив състав във времето. Последният ще се променя вследствие на излизане на задачи от цикъла и влизане на други, които ще ги заместват.

Ходът на изпълнението на всяка отделна задача зависи от текущата стойност на условието за край *CEC*, която тя генерира. Ето защо е важно да се анализират събитията в двата случая на това условие, както по отношение на изходната, така и по отношение на входната точки. Приема се, че влезлите в тялото на цикъла задачи 1, 2 и 3 са изпълнили няколко повторения и се намират в положението, показано на фиг.6, в най-лявата колонка.



Фиг. 6. *Процес на излизане от цикъла*

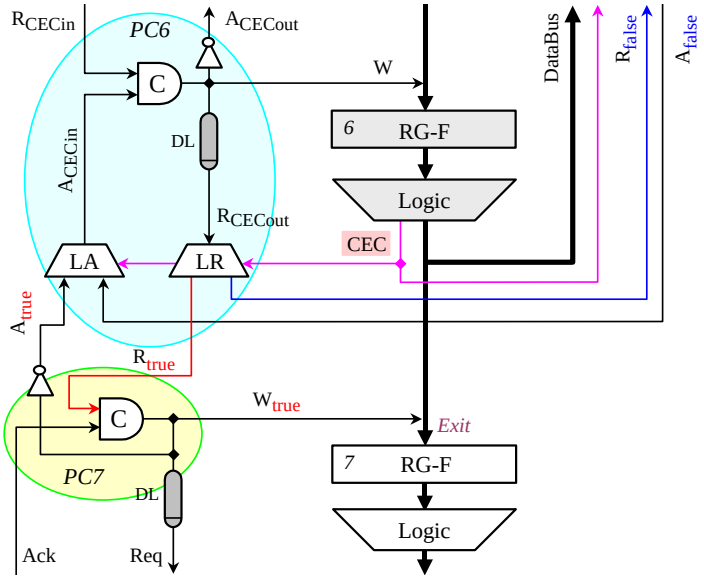
Нека за задача 1 звено 6 (фиг.2) е генерирало условието *CEC*=0. В резултат на това тя преминава по обратната връзка в звено 3. В последствие задачите последователно слизат от звено на звено, като в третата колонка се вижда, че звено 3 отново е свободно. В звено 6 се намира задача 2, за която *CEC*=1. Задача 2 излиза от цикъла, а в него влиза нова задача 4. Задача 3 се превърта, а задача 1 излиза от цикъла. Така в края на рисунката в тялото на цикъла остава да се превъртва групата от задачи (4, 3, 5). Изложеният току що анализ изявява още един нов проблем. Същността на този проблем се състои в реда на излизащите от цикъла задачи. Тъй като броят на повторенията не е известен и е индивидуален за всяка задача, не може да се очаква редът при зареждането им да се повтори при излизането им от цикъла. Така в отделните моменти съставът на групата в цикъла се променя. В заключение на казаното следва да бъде формулирана за този раздел нова четвърта задача, състояща се във възстановяване (при необходимост) на реда на слизащите от конвейера задачи. Тази задача вече има решение [4] и няма да бъде дискутирана тук. По-долу са представени решенията на формулираните задачи, които позволяват създаването на метод за асинхронна микроконвейерна реализация на алгоритмични структури от вида цикъл.

7. Конвейерен автомат на звеното с условие за край на цикъл

Конвейерният автомат на звеното, което генерира условието за край на цикъла *CEC* (изходното звено), по същество е свързан са два други конвейерни автомата. Единият автомат управлява звеното, което се намира по изхода от цикъла (звено 7, фиг.2). В това звено попадат резултатите, получени от циклическите изчисления, когато условието за край бъде изпълнено *CEC*=1. По същество, това е изходът "*истина*" (*CEC=true*). Другият автомат е този, който управлява звеното във входната точка на цикъла. Връзката с него се осъществява по изхода "*лъжа*" на условния преход (*CEC=false*).

7.1. Конвейерен автомат с 2-фазов протокол за трансфер на данни

Синтезираната логическа структура на конвейерен автомат *PC6* с 2-фазов протокол, управляващ звеното, което генерира условието за край на цикъла, е представена на фиг. 7.



Фиг. 7. Логическа структура на конвейерен автомат с условие за край на цикъл

Разглеждано като част от логическата структура на конвейера, микроконвейерното звено 6, което генерира условието за край на цикъла *CEC*, независимо какъв е той по вид, има аналогичните проблеми на всяко друго звено, генериращо условие за преход. По тази причина структурното решение за автомата на разглежданото тук звено наподобява решението, което е изложено в [2].

Както се вижда от фигурата, сигналът потвърждение A_{CECin} е функция, реализирана от схемата LA (*Logic Acknowledgement*) от потвържденията A_{true} и A_{false}, а двете заявки R_{true} и R_{false} се формират от сигнала R_{CECout} в схемата LR (*Logic Request*).

7.2. Синтез на сигнал потвърждение

Превключването на конвейерния автомат в последното звено от тялото на цикъла е функция от два входни сигнала: A_{true} и A_{false}. Последните оповестяват готовността на всеки от клоновете по отделно, т.е. те са "родители" на сигнала A_{CECin}. Поради значително по-голямата сумарна латентност на циклическата структура като цяло, в сравнение с тази на микроконвейерното звено, намиращо се след изхода от цикъла, може да приемем, че по време на циклическите изчисления, потвърждението A_{true} е постоянно подадено и очаква момента, когато ще бъде използвано. Потвърждението A_{true} идва от звено 7, което се намира

след циклическата структура, което поради закъснението в последната се предполага, че вече е завършило своето изчисление. До този момент обаче актуалното потвърждение идва от звеното в клона "лъжа", а то е A_{false}. Това е така по причина на многократните завъртания на цикъла по обратната връзка. В тези условия, актуалното потвърждение пристига в отговор на заявката R_{false} в момент, когато логическата стойност на условието *CEC* е стабилно установена във времето. От тук следва, че логическата схема LA ще осъществява само мултиплексиране на двете входни потвърждения, според следната логика.

$$A_{CECin} = A_{true} \cap CEC \quad \cup \quad A_{false} \cap \overline{CEC} \ . \tag{2}$$

Сигналът A_{CECin} ще има стойността на идващото от съответния клон потвърждение, в следствие на което не е необходимо конвертиране на тази стойност.

Следва да бъде отчетен и събитийния характер на новите стойности. Моментът на поява на новата логическа стойност на условието за преход *CEC* и моментът на поява на заявката R_{CECout}, се състезават във времето. Резултатът от това състезание е или равен или се печели от *CEC*. Мултиплексорът на потвържденията LA, който ще се управлява от условието *CEC*, трябва да се превключи не в момента на възникване на новата стойност на условието *CEC*, а в момента на появата на заявката R_{CECout}. Това налага състезателността да бъде отстранена, което можем да постигнем единствено с помощта на тригер. В момента на възникване на заявката R_{CECout} този тригер трябва да фиксира стойността на условието *CEC* и да я поддържа до следващия такт. Така той ще има още една положителна роля – ще предпазва мултиплексора LA от нежелани превключвания по време на самото изчисление на следващата нова стойност на *CEC*.

7.3. Генериране на заявки към разклоненията

Конвейерният автомат, управляващ звеното с условен преход, разпространява към следващите звена заявка, отбелязана като R_{CECout}. Заявката не може да се подаде директно към входовете на конвейерните автомати в началото на всеки от алтернативните клонове. Съответните запитвания, които следва да получат конвейерните автомати, са означени с R_{true} за клона "истина" и с R_{false} за клона "лъжа" и са функция на логическата схема LR (фиг.7). Непосредственото подключване на заявката R_{CECout} не е възможно, защото в точката на разклонението тя влиза в сложна функционална връзка с логическата стойност на условието *CEC* от една страна, и с текущото (завареното) състояние на конвейерния автомат в началото на всеки клон, от друга страна. Това пък се налага от вида на самите автомати, които както вече беше споменато, използват 2-фазов протокол за управление на трансфера. Последното означава, че всяко тяхно превключване ($0\rightarrow1$ и $1\rightarrow0$) причинява запис в регистрите фиксатори и старт на изчисленията в звената. Ако се разглежда превключване в началото на клон от микроконвейера, то освен логическата стойност на условието за преход *CEC* (0 или 1) следва да се отчете и завареното състояние на съответния автомат. Ще отбележим, че C-елементът в схемата на автомата се превключва както при съчетаване на две входни единици, така и на две нули. С други думи сигналите R_{true} и R_{false} са функции не само от превключването на сигнала R_{CECout}, но и от стойността на условието *CEC*, а така също и от състоянието на конвейерните автомати на входа на разклоненията. Например, ако стойността на условието за преход е нула (*CEC*=0), това означава, че изчисленията трябва да продължат в клона "лъжа", т.е. в звено 3, като се затвори за пореден път обратната връзка на цикъла. Единственото, което различава този тип разклонение от обикновеното разклонение, е точно тази няколкократна повторяемост на ситуацията, представляваща същността на структурата цикъл. Ако състоянието на автомата *PC3* в този клон, в който ни насочва условието, е

единица, което се поддържа във времето до този момент от стойността R_{false}=1, то той трябва да се превключи в ново състояние нула, за да стартира чрез заден фронт на сигнала *W* тези изчисления. За да стане това, на входа на този С-елемент трябва да се съчетаят две нули. Не трябва да се забравя и факта, че всяка нова стойност на сигнала R_{CECout} (и 0 и 1) представлява по същество нова заявка, излизаща от автомата *PC6* на звено 6.

Логиката, която току що беше пояснена, е изразена чрез таблиците на истинност 1 и 2, в които сигналите W_{true} и *W* представят състоянието на съответните С-елементи. Тук следва допълнително пояснение. За разлика от случая на обикновено разклонение, където се използваха два сигнала W_{true} и W_{false}, тук вторият сигнал е означен само с *W*, т.е. без подсказката *fasle*. Това е така, защото звеното във входната точка не е обикновено начално звено на клон в алгоритъм. В нашият случай входното звено е звено в обща точка. То записва нови данни както в качеството си на начално звено в клон "лъжа", така и като обикновено звено при първо навлизане на задачи в циклическата структура, ето защо сигналът за запис на това звено е означен само *W*.

Таблица 1. Заявка към конвейерен автомат в клон "лъжа"

CEC	*W*	R_{CECout}	R_{false}
0	0	появява се заден фронт 0	1 , превключва се
0	0	появява се преден фронт 1	1 , превключва се
0	1	появява се заден фронт 0	0 , превключва се
0	1	появява се преден фронт 1	0 , превключва се
1	0	появява се заден фронт 0	0, не се превключва
1	0	появява се преден фронт 1	0, не се превключва
1	1	появява се заден фронт 0	1, не се превключва
1	1	появява се преден фронт 1	1, не се превключва

Таблица 2. Заявка към конвейерен автомат в клон "истина"

CEC	W_{true}	R_{CECout}	R_{true}
0	0	появява се заден фронт 0	0, не се превключва
0	0	появява се преден фронт 1	0, не се превключва
0	1	появява се заден фронт 0	1, не се превключва
0	1	появява се преден фронт 1	1, не се превключва
1	0	появява се заден фронт 0	1 , превключва се
1	0	появява се преден фронт 1	1 , превключва се
1	1	появява се заден фронт 0	0 , превключва се
1	1	появява се преден фронт 1	0 , превключва се

Въз основа на горните таблици на истинност са синтезирани следните логически функции:

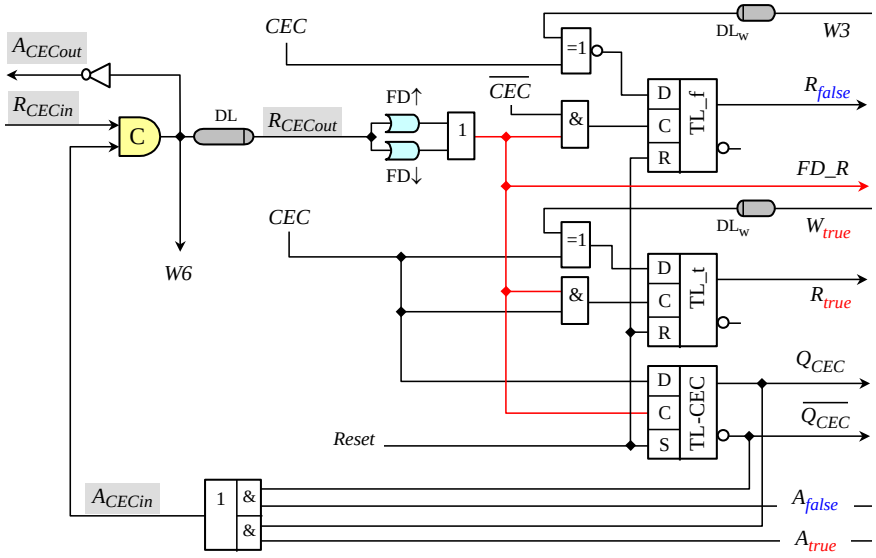
$$R_{true} = CEC \oplus W_{true} \ . \tag{3}$$

$$R_{false} = \overline{CEC \oplus W} \ . \tag{4}$$

Както се вижда, логиката на заявките R_{true} и R_{false} не зависи от заявката R_{CECout}, което се очакваше. Зависимостта на заявките R_{true} и R_{false} не е от стойността на сигнала R_{CECout}, а от времето, т.е. от моментите на неговото превключване. Следва да се разбира, че смяната на стойността на R_{CECout}, т.е. появата на негов фронт, маркира момента във времето, в който операционната логика на това звено завършва изчисленията си. Този момент не е задължително да съвпада с появата на истинната стойност на условието *CEC*. В зависимост от сложността на изчислението на *CEC*, в общия случай следва да се приема, че истинната стойност на *CEC* може да се появи и по-рано във времето спрямо новия фронт на сигнала R_{CECout} или най-късно едновременно с него и никога по-късно от него. В една непосредствена реализация на (3) и (4), по-ранното явяване на *CEC* ще доведе до по-ранно формиране на стойностите на заявките R_{true} или R_{false}, а от там и до по-ранно стартиране на съответния клон на конвейера. Това стартиране ще започне със запис на данни в регистъра фиксатор, а те в общият случай все още няма да са достигнали във времето истинните си стойности. Така най-вероятно изчисленията могат да стартират с грешни данни.

Основният извод, който се налага от така изложените съображения е, че формулите (3) и (4) определят стойностите на заявките, но моментът, в който те следва да се появят и да подействат, се определя от момента на превключване на сигнала R_{CECout}. Или казано по друг начин, новите стойности на заявките R_{true} или R_{false} трябва да се явят в отговор на фронт в сигнала R_{CECout}. Последното означава, че формирането на заявките R_{true} и R_{false}, не е възможно да бъде постигнато единствено с комбинационна логика.

Изложените разсъждения доказват, че времевата зависимост на изчислените стойности (3) и (4) може да се реализира само с помощта на запомнящ елемент – тригер. Тъй като запис трябва да се извършва при всяко превключване на С-елемента, синхронизиращият тригер трябва да бъде от тип DEDTFF (D-тригер, работещ и по двата фронта). В окончателната логическа схема на конвейерния автомат *PC6* (фиг.8) е представено едно предпочетено тук решение, основаващо се на обикновен D-*Latch* тригер и два детектора на фронт – FD↑ за преден и FD↓ за заден. Логическата схема, която беше отбелязана с LR във фиг.7, съдържа С-елемента на звеното с условен преход, схемата LA, обединяваща потвържденията A_{true} и A_{false} , както и двете тригерни схеми, генериращи заявките R_{true} и R_{false}.



Фиг. 8. Логическа схема на 2-фазов конвейерен автомат за звено с условие за край на цикъл

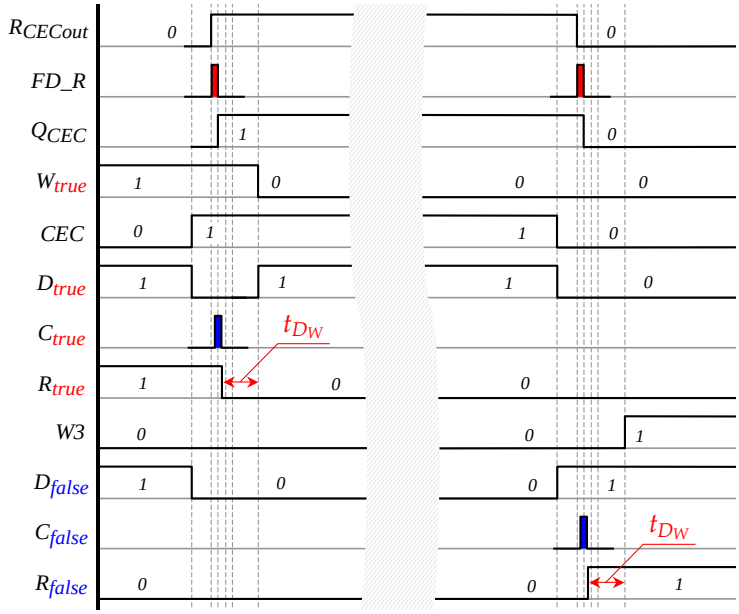
Както се вижда от схемата, пулс-генераторите се обединяват в схема ИЛИ, чийто изход е означен *FD_R*. Този сигнал реализират запис по входа С на съответния тригер – в тригер TL-CEC винаги, а в другите два тригера, според стойността на условието за край.

Изчислената според (3) логическа стойност на заявката R_{true} постъпва по входа D и се съхранява в тригера до следващия път, когато ще бъде избран същия клон. Сигналът *Reset* е необходим в началния момент, в който всички конвейерни автомати принудително се установяват в изходно състояние. Аналогична логическа схема формира според (4) заявката R_{false} към конвейерния автомат в клон "лъжа".

Тъй като в отговор на изпратената заявка R_{true} или R_{false} съответният конвейерен автомат ще се превключи и ще върне по обратната връзка нова стойност на сигнала *W*, която застрашава надеждността на записа в D-тригера TL_t или TL_f, то това налага задържане във времето, докато изчезне импулсът за запис от входа C. Закъснението се осигурява от елемента DL_W, което се осигурява конструктивно от неравенството $t_{DLw}>t_{FD}$.

В схемата е включен тригер TL_CEC, който има за задача да запомни логическата стойност на условието за край на цикъла *CEC* в момента, когато звеното завърши своите изчисления, т.е. когато се появи заявката R_{CECout}. Така той няма да допусне новата стойност на условието *CEC* да манипулира преждевременно мултиплексора на потвържденията, поддържайки стабилна текущата стойност до края на следващото изчисление.

Изходното състояние на тригерите TL_t и TL_f е без значение, но е прието те да се нулират от сигнал *Reset*. В същото време, принудителното състояние на тригер TL_CEC е единица, което се изисква от конвейерния автомат на звеното във входната точка. Това положение е пояснено чрез времедиаграмата от фиг.9.



Фиг. 9. Две възможни превключвания на автомата на изходното звено

Времедиаграмата изобразява две превключвания на автомата на изходното звено. Първото е при стойност на условието за край *CEC*=1. Изобразените стойности до този момент на сигналите W_{true} и *W* са приети за единица и нула съответно, като възможни. Вижда се, че стойността на заявката R_{false} и състоянието на автомата *W3* в клона "лъжа" не се променят. В същото време заявката към клон "истина" се превключва. В резултат на това се превключва и съответният автомат – сигналът W_{true}=0. Новото му състояние е отразено със закъснение t_{Dw}.

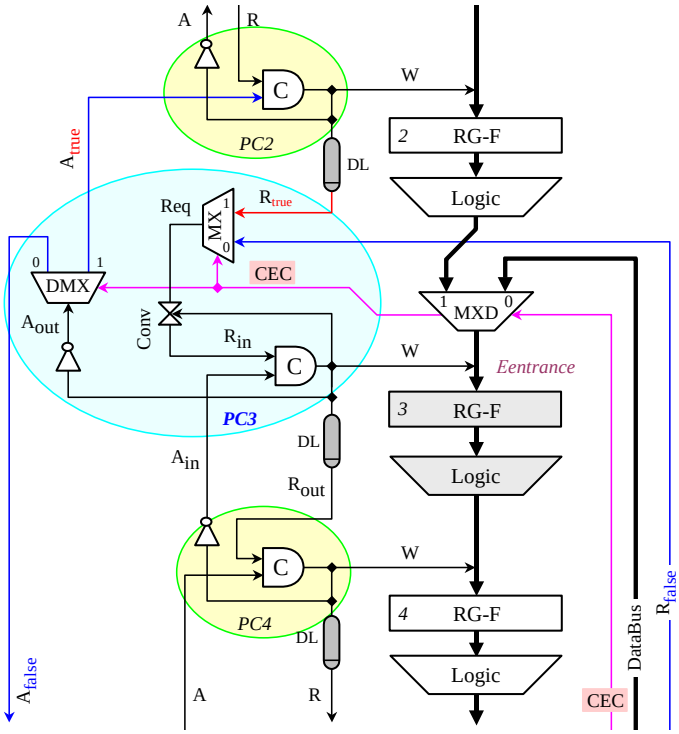
Второто превключване е при стойност на условието за край *CEC*=0 и следователно актуалната заявка ще бъде насочена в клона "лъжа". Така заявката R_{false} се превключва в единица. В резултат на това автоматът на звено 3 също се превключва (*W3*=1), което е отразено във времедиаграмата със същото закъснение t_{Dw}.

8. Конвейерен автомат на звеното във входната точка

Общата характеристика на звеното във входната точка на цикъла беше изложена в началото. Данновите входове на регистъра фиксатор RG-F в това звено са два. Те трябва да се мултиплексират в зависимост от логическата стойност на условието за край на цикъла *CEC*. Като се има предвид, че тази стойност се съхранява в тригер TL_CEC, който е част от схемата на автомата, управляващ изходното звено, то неговото състояние може да се използва и в схемата на автомата на звеното във входната точка. Това е така, защото съответното зареждане с данни на това звено става по времето на актуалната стойност на условието *CEC*.

8.1. Конвейерен автомат с 2-фазов протокол за трансфер

От фиг. 2 се вижда, че автоматът, който управлява звеното в тази точка, е свързан със следващото звено традиционно чрез сигналите *Request* (R_{out}) и *Acknowledgement* (A_{in}). Двустранна е обаче връзката му с автоматите на предходни звена. Фиг.10 представя синтезираната за този автомат логическа структура.



Фиг. 10. Логическа структура на конвейерен автомат във входна точка на цикъл

Генериране на заявка

Заявката, която трябва да получи автоматът на звеното във входната точка на цикъла, е означена R_{in}. Схемата на конвейерния автомат *PC3*, който управлява звеното във входната точка на цикъла, съдържа мултиплексора MX. Той обединява двете заявки към това звено, означени R_{true} и R_{false}. Мултиплексорът MX се управлява от условието за край на цикъла *CEC*, което се поддържа стабилно във времето от тригер TL-CEC (фигура 3.8.8). Това означава, че мултиплексорът ще бъде правилно превключен както при първо влизане в цикъла, така и при всяко завъртане на изчислителния процес по обратната връзка. Това е така, защото след всяко излизане от цикъла, което се реализира при стойност на условието за

преход *CEC*=1, тригер TL-CEC остава в единично състояние. Именно това състояние държи конвейерния автомат на звеното във входната точка включен към предходния автомат *PC2*, т.е. изгражда връзката $Req=R_{true}$. Това състояние обяснява и защо сигнал *Reset* е подаден към S входа на тригер TL-CEC. Алтернативната връзка $Req=R_{false}$ се поддържа по време на циклическите повторения, когато *CEC*=0. Същото важи и за мултиплексора на данновите шини MXD.

Логическата стойност на заявката *Req*, такава каквато излиза от мултиплексора MX, не винаги е тази, която е актуална за Мюлер С-елемента, на чийто вход R_{in} следва да бъде подадена. Това неудобство се дължи на 2-фазовия протокол на автомата, който разглеждаме. Според него всяко състояние на С-елемента е работно и то трябва да се изменя на противоположното при всеки старт на микроконвейерното звено. Това означава, че ако С-елементът е в състояние нула (*W*=0), то при нова заявка той трябва да се превключи в състояние единица. Това е възможно само ако тази заявка се появи като единица. Аналогично е положението за обратното превключване.

Възможното несъответствие между логическата стойност на новата заявка R_{out} и очакваната стойност R_{in} налага нейното преобразуване. Така тази стойност ще бъде съобразена със завареното състояние на С-елемента в автомат *PC3*. За целта съставяме таблица на истинност.

Таблица 3. Логическа стойност на заявката R_{in} и потвърждението A_{out}

Заявката *Req* пристига с нова стойност:	С-елементът е заварен в състояние:	С-елементът трябва да се превключи в състояние:	Следователно R_{in} трябва да има стойността:	Трябва да бъде върнато потвърждение A_{out} със стойност:
0	0	1	1 ($\overline{Req}$)	1
0	1	0	0 (Req)	1
1	0	1	1 (Req)	0
1	1	0	0 ($\overline{Req}$)	0

От таблицата се вижда, че когато логическата стойност на новата заявка съвпада със състоянието на С-елемента (първи и четвърти ред), необходима е инверсната стойност, а когато са различни – необходимата стойност е същата. Тази логика

$$R_{in} = \overline{(Req \oplus W)} \cap \overline{Req} \ \cup \ (Req \oplus W) \cap Req = \overline{W} \qquad (5)$$

изразява формално схемата на конвертора Conv (фиг.10). Според (5) входната заявка има пряка връзка с изходното състояние, но тя следва да се осъществява и актуализира само в момента, в който се появява стойността на условието *CEC*. Този момент се бележи от фронтовете на превключване на заявката R_{CECout} (фиг.8), които са реализирани от сигнала, означен *FD_R*. Казаното означава, че връзката (5) трябва да се осъществи чрез тригер, т.е. заявката R_{in} трябва да се подаде към С-елемента в нужния момент.

Синтез на сигнал потвърждение

След като конвейерният автомат на звеното във входната точка се превключи, той трябва да върне сигнал за потвърждение A_{out} към звеното, от което е получил заявката. Звеното е било определено от стойността на условието *CEC*, следователно пак то определя направлението на актуалното потвърждение. За целта в схемата е включен демултиплексорът DMX. Така, когато *CEC*=0, стойността A_{out} ще бъде насочена към автомата в клона “лъжа”,

т.е. $A_{false} \equiv A_{out}$. В обратния случай, когато CEC=1, стойността A_{out} ще бъде насочена към автомата в клона “истина”, т.е. $A_{true} \equiv A_{out}$.

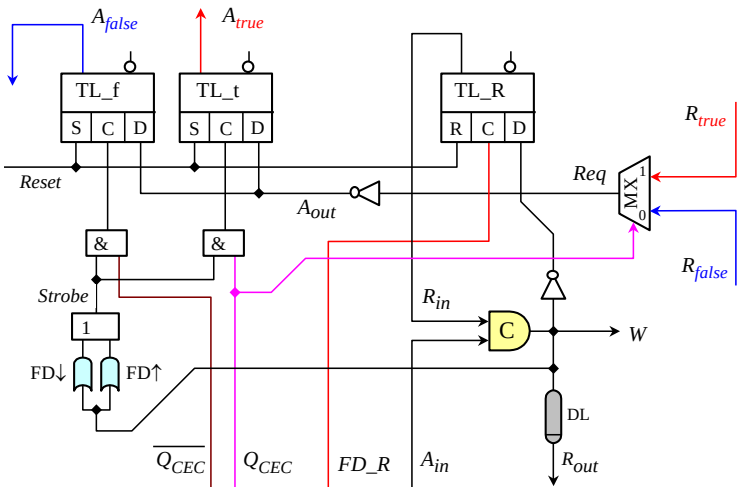
При чистото демултипликсиране на сигнал A_{out} обаче, към неактуалния клон винаги ще се изпраща нова стойност нула. От една страна тази нула може да предизвика неправомерно превключване на конвейерния автомат в този неактуален клон, а от друга страна, тъй като клонът е неактуален, то стойността на потвърждението към него трябва да остава неизменна се във времето, докато той чака обръщение. Последното означава, че стойността на върнатото преди това актуално потвърждение следва да бъде запомнена и поддържана във времето от тригер. Този тригер трябва да променя състоянието си в съответствие с новата стойност на сигнала потвърждение, само когато клонът за който той работи, бъде определен за актуален. С други думи тригерите с това предназначение трябва да са на брой два.

Следващата задача, която трябва да бъде решена, е каква трябва да бъде стойността на сигнала потвърждение. Тъй като конвейерният автомат *PC3* на звеното във входната точка би могъл преди този такт да се е превключвал неизвестен брой пъти, то връщаната от него в текущия такт стойност A_{out} може да не съответства на очакваната от автомат *PC2* или от автомат *PC6* (фиг.2). Налага се преобразуване на стойността на сигнала A_{out} в необходимата, което се дължи на 2-фазовия протокол за управление на данновия трансфер.

Логиката за определяне на актуалната стойност на потвърждението се основава на логиката, по която се превключва Мюлер С-елементът, а именно (вижте например първи ред в таблица 3), когато автоматът *PC3* се е превключил в резултат на получена заявка със стойност нула, към автомата, подал тази заявка трябва да бъде върнато потвърждение със стойност единица. Изказаната логика се съдържа в таблица 3, а решението е

$$A_{out} = \overline{Req} \tag{6}$$

Окончателната принципна логическа схема на конвейерния автомат на звеното във входната точка е представена на фиг.11.



Фиг. 11. Логическа схема на 2-фазов конвейерен автомат за входно звено на цикъл

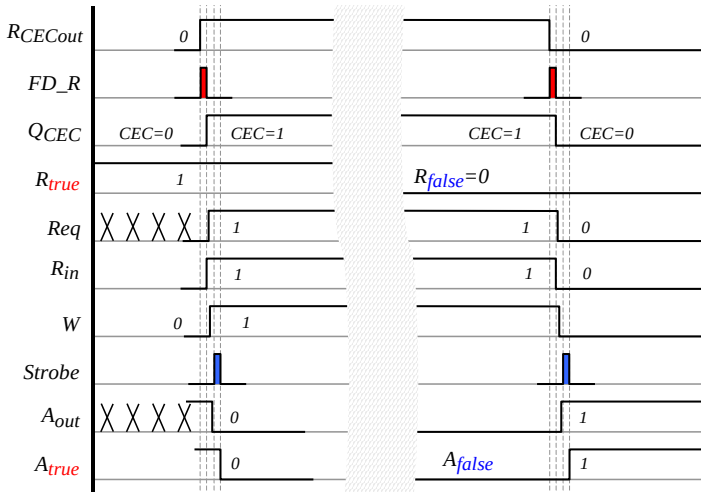
Схемата реализира функциите (5) и (6) като отчита времевата им зависимост от събитията, които ги формират. Така стойностите на потвържденията A_{false} и A_{true} се записват в съответния тригер, когато се превключи С-елементът на автомата, а заявката R_{in}, която той получава, се записва в тригер TL_R от сигнал *FD_R*. Изходното състояние на автомата определя сигнал *Reset*.

Функционирането на автомата на звеното във входната точка е илюстрирано с времедиаграмата от фиг. 12, при следните начални условия: нека при предходното

превключване автоматът да е записал данни по обратната връзка. Това означава, че условието за край е било *CEC*=0, откъдето следва, че в тригер TL_R има записана нула, т.е. R_{in}=0. Следва, че С-елементът е в състояние 0 (*W*=0).

Прието е, че при новото превключване автоматът ще запише данни по правата връзка, тъй като ще приемем още, че условието за край ще бъде *CEC*=1. Прието е също, че заявката от предходното звено 2 има стойност R_{true}=1. Последното означава, че тригер TL_t поддържа стойност 1, т.е. A_{true}=1. Новото превключване започва с превключването на заявката R_{CECout} в изходното звено, което ще приемем за единица.

На времедиаграмата, след превключването на заявката R_{CECout} в единица, е изобразено следващото й превключване, което е в нула. При това превключване е прието, че стойността на условието за край на цикъла е *CEC*=0, което означава, че този път входното звено ще приеме данни по обратната връзка. Вижда се, че превключването зависи от стойността на заявката R_{false}, която е приета да е нула (R_{in}=R_{false}=0). Тази стойност е поддържана във времето до този момент от стойността A_{false}=0. След превключването на автомата (*W*=0), в клона "лъжа" към автомата на изходното звено се връща потвърждението A_{false}=1.



Фиг. 12. Две последователни превключвания на автомата на входното звено

9. Задача трета

По-горе в раздел 5 беше изявена задача, която получи пореден номер три. Тя изисква, в началните тактове, при първоначално зареждане на конвейера със задачи, обратната връзка на циклическото тяло да бъде изключена. При това, данновите връзки на входното звено на тялото на цикъла трябва да се поддържат включени към предходното звено, с което се осигурява зареждане на многостепенното тяло с максимално допустимия брой задачи. Беше определено, че ако степените в конвейеризираното тяло са *m* на брой, то задачите трябва да бъдат (*m-1*) на брой. Имат се предвид регистри фиксатори от тип *Latch*. Обратната връзка трябва да бъде включена веднага след зареждане на този брой задачи. От този момент нататък данновите връзки в изходната и входната точки трябва да се управляват от условието за край на цикъла *CEC*. В началния момент, за който важи трета задача, протича процес на зареждане на циклическото тяло със задачи, който е илюстриран с фиг.3. Както е показано на фигурата, обратната връзка трябва да бъде включена точно в такт (k+7). Заедно с това управлението на входния мултиплексор в този момент трябва да се преотстъпи на условието за карай на цикъла *CEC*. Процесът на зареждане на циклическото тяло със задачи е съпроводен с изчислителен процес, който всяко звено провежда върху заредените в него данни. Анализира се такт (k+5) от същата фигура. В този момент задача 1 постъпва в

изходното звено 6 и стартира своите изчисления. Интересното за тях е това, че те ще получат първата стойност на условието за край на цикъла *CEC*. Докато това условие се изчислява, в посока на входната точка се изкачва свободното звено, което в момент (k+7) се оказва в звено 3. Изчислението в звено 6 има латентност t6. В същото време, входното звено ще бъде свободно след време (t5+t4+t3), което е сумарната латентност на останалите предходни звена. След като звено 6 приеме данни от звено 5 и му върне сигнал потвърждение, последното прави същото, т.е. може да се приеме с първо приближение, че тези две съседни звена (6 и 5) стартират своите изчисления почти едновременно. Това ще позволи да се илюстрира казаното по-горе със следната рисунка, позволяваща да се сравнят възможните времеви закъснения

Тъй като в общия случай латентностите на звената при конкретните изчисления не са известни, а така също и броят на степените в конвейера не е известен, както и структурата му не е известна, то са възможни следните отношения на латентността на изходното звено t_m (с m беше означен броя на звената в тялото на цикъла)

$$\begin{cases} t_m < \sum_{k=1}^{m-1} t_k \ ; \\ t_m \geq \sum_{k=1}^{m-1} t_k \ . \end{cases} \qquad (7)$$

Първото отношение съответства на горната рисунка. За изчислителния процес това означава, че изходното звено е завършило своите изчисления и стойността на условието *CEC* е готова. Предхождащата структура от звена обаче не е завършила, а може би дори не е заредена до край със задачи. Това означава, че изходното звено ще остане в състояние на очакване, докато настъпи момента, в който ще се свърже с входното звено, от което ще получи сигнал потвърждение и където ще може да управлява входния даннов мултиплексор. Разрешение за това ще му даде решението на трета задача, което предстои да представим.

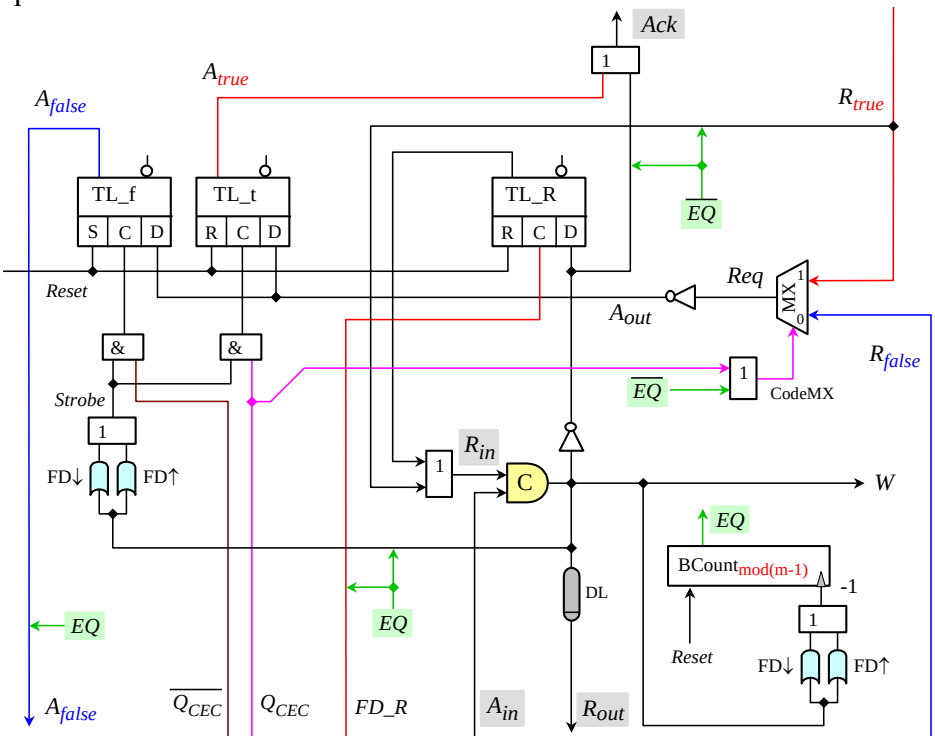
Второто отношение на (7) в общия случай е малко вероятно. Според него предходната структура от звена е заредена и е завършила изчисленията си, като очаква управление на обратната връзка от изходното звено. С други думи принудителното управление на входната точка е завършило и е прекратено.

Направеният анализ показва, че във всички случаи, при първоначално зареждане на конвейерното тяло на цикъла, изходното звено все още не е в състояние да управлява автомата на входното звено и процесът на зареждане трябва да се контролира от друг елемент. Този елемент трябва да бъде двоичен брояч по модул (*m-1*), за да може да отброи необходимия брой заредени в тялото на цикъла задачи. Върху логическата схема от фиг.11 е проведен допълнителен синтез, в резултат на което е получена окончателната принципна логическа схема на конвейерния автомат, който управлява звеното във входната точка на цикъла, представена на фиг.13. В схемата се вижда споменатият брояч BCount, който е декрементен. Началното му съдържание е числото (*m-1*), което се установява по сигнал *Reset*. Нулевото съдържание на брояча се разпознава от дешифратор, който генерира единица на изхода *EQ*. Стойността на този сигнал се използва за управление на посочените в схемата връзки. Правата фаза на сигнала *EQ* забранява разпространението на сигналите A_{false} , *FD_R* и *W* в показаните направления. Докато броячът не се нулира, сигналът има стойност нула

(EQ=0). Инверсната фаза на сигнала EQ забранява разпространението на сигналите R_{true} и $\overline{W}$. Мултиплексорът MX се управлява от функцията $CodeMX$

$$CodeMX = CEC \cup \overline{EQ} \ . \qquad (8)$$

Тази логическа функция изчислява стойност единица винаги, когато съдържанието на брояча е различно от нула, т.е. по време на зареждането на тялото на цикъла със задачи. Докато тази функция има стойност единица, мултиплексорът MX поддържа вход 1. Така заявката R_{true} достига входа R_{in} на Мюлер С-елемента. Същата функция трябва да управлява и данновия мултиплексор MXD (фиг.10). Докато броячът не се нулира, автоматът на звеното във входната точка комуникира с предходното звено 2 (сигнали Ack и R_{true}) и със следващото звено 4 (сигнали A_{in} и R_{out}) от тялото на цикъла. Тригерите TL_f, TL_t и TL_R не се превключват и остават в изходно състояние, определено от сигнал $Reset$. Така, тъй като автоматът на изходното звено 6 ще получава стойност A_{false}=0, то ще остава в изходно състояние. В момента, в който броячът се нулира, сигналът EQ получава стойност единица. Забранените до момента връзки се разрешават, а разрешените се забраняват. След достигане на съдържание нула броячът не трябва да функционира. Това означава, че той не трябва да бъде кръгов брояч.



Фиг. 13. Окончателна принципна логическа схема на 2-фазов ковейерен автомат за входно звено на цикъл

След началното зареждане на задачи в тялото на цикъла, когато задача 1 завърши своите изчисления в звено 5, тя ще поиска да се прехвърли в последното звено 6, подавайки заявка към автомата му. Този автомат няма да може да удовлетвори тази заявка, тъй като достъпът на потвърждението A_{false} е блокиран от сигнала EQ=0, въпреки, че тригер TL_f е в единично състояние.

Докато не бъде заредено звеното във входната точка с последната (m-1)-ва за тялото на цикъла задача, забраната от сигнала EQ ще е в сила. Тази забрана ще бъде снета в момента, в който звеното бъде заредено с последната задача, тъй като тогава броячът ще се нулира. С появата на сигнал EQ=1 задача 1 ще премине в последното звено и ще започне изчислението

на първата стойност на условието *CEC*. През това време следващите задачи ще слязат надолу и когато звеното във входната точка се освободи, в него ще постъпи задача, която ще определи условието *CEC*. От този момент нататък схемата от фиг. 13 ще функционира аналогично на схемата от фиг. 11.

10. Заключение

В смисъла на това изследване не са открити сходни публикации. Многотактовите циклически структури, изследвани тук, не винаги могат да бъдат разгънати и така тяхната реализация не може да бъде да бъде по-проста. Ето защо това изследване е актуално.

Както беше посочено по-горе, представеното комплексно решение не зависи от вида на цикъла. В хода на анализа бяха изявени няколко нови задачи, решението на една от които е представено в предходна публикация. Получените решения на останалите задачи са приложими в микроконвейери, използващи конвейерни автомати както с 2-фазов протокол за трансфер на данни, така и с 4-фазов.

Приложението на асинхронната конвейерна реализация върху циклически структури води до значително повишаване на производителността на подобни изчислители. В циклическото тялото могат да съществуват достатъчно сложни и общи по вид алгоритмични последователности.

Наличието на решения за конвейерните автомати в изходната и входната точки на циклическото тяло в конвейера, които не зависят от броя на степените в циклическото тяло, както и от вида на микроконвейерните звена, по същество създава метод за хардуерна реализация на такива общи алгоритмични структури.

От друга гледна точка, получените решения създават възможност за изграждане на универсална структура. Конвейерното тяло на тази структура може лесно да бъде подменяно. Това е възможно дори в процеса на реално функциониране. Възможността структурата да бъде реконфигуруема, представлява едно изключително полезно следствие, което предстои да бъде изследвано.

Литература:

[1]. Tyanev, D.S., Four-phase micro-pipeline with one-cycle and multi-cycle micro-pipeline sections, Computer Science and Technologies, Publication of Computing and Automation Faculty Technical University of Varna, Bulgaria, ISSN 1312-3335, VII 1/2009, pp. 3-12.

[2]. Tyanev, D.S., Branch management in asynchronous micro-pipeline, Computer Science and Technologies, Publication of Computing and Automation Faculty Technical University of Varna, Bulgaria, ISSN 1312-3335, VII 2/2009, pp. 3-12.

[3]. Tyanev, D.S., S.I. Kolev, D.V. Yanev, Race Condition free Asynchronous Micro-Pipeline Units, International Conference on Computer Systems and Technologies – CompSysTech'10, 17-18 June 2010, Sofia, Bulgaria.

[4]. Tyanev, D.S., V.T. Bozhikova, S.K. Gerganov, P.G. Georgiev, Algorithm for micropipeline buffer control, Applied Technologies and Innovations, ISSN 1804-1191, vol. 4, Issue 1, April 2011, pp.12-21.

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

СИМУЛАТОР НА УЧЕБЕН МИКРОКОМПЮТЪР С ОПРОСТЕНА АРХИТЕКТУРА

Станимир С. Станев

Резюме: За подпомагане изучаването на архитектурата на съвременни процесори и програмирането на Асемблер за тях, е разработен симулатор за хипотетичен елементарен учебен микрокомпютър – СЕКТ, с опростена последователна архитектура и едноадресни инструкции. Съставена е система от 20 машинни инструкции, които изпълнява симулаторът на микрокомпютъра. Показан е пример от реализация на симулации на елементарна асемблерова програма и са посочени перспективи за усъвършенствуване на симулатора с цел постигане на по- добър учебен ефект. Симулаторът с успех се използва в обучението на студенти от ФМИ на Шуменския Университет „Епископ Константин Преславски" и в още две учебни заведения в страната.
Ключови думи: Компютърна архитектура, програмиране на Асемблер, компютърни симулации, микрокомпютри.

Simulator of a training microcomputer with simplified architecture

Stanimir S. Stanev

Abstract: Software simulator of hypothetical elementary training microcomputer – SEKT, with simplified sequential architecture and single address instructions is developed, to support the study of the architecture of modern processors and assembly languages. A system is composed of 20 machine instructions executed by the simulator of the microcomputer. An example of the realization of simulations of elementary assembly program is shown. The prospects for improving the simulator to achieve better learning effect are discussed. The simulator is used with success in educating students at Bishop Konstantin Preslavsky University of Shumen and in two higher schools in the country as well.
Keywords: computer architecture, assembly language, computer simulations, microcomputers.

1. Увод

Студентите, които не са от инженерни компютърни специалности, но изучават дисциплините „Компютърни архитектури" и „Компютърни системи" (това обикновено са студентите в специалности „Информатика" и „Математика и информатика" от класическите университети), срещат определени трудности при усвояване на архитектурата на съвременните компютри и програмирането на ниско ниво за микропроцесори. Опитът от преподаването на основите на функциониране на компютри с последователна архитектура показва, че добър методически подход при преподаване на програмирането на Асемблер за Intel - базирани микропроцесори е първоначалното запознаване на обучаемите с архитектурата на опростен микрокомпютър с минимален брой инструкции, и след това преход към изучаване на програмирането на Асемблер за стандартните микропроцесори. Известни са редица учебни емулатори и софтуерни симулатори [1,2], но използването на пълните им възможности не е възможно безплатно. Това определя необходимостта от разработенване на софтуерен симулатор на елементарен микрокомпютър, който да е лесен за усвояване и да работи на широко разпространени Windows - базирани платформи.

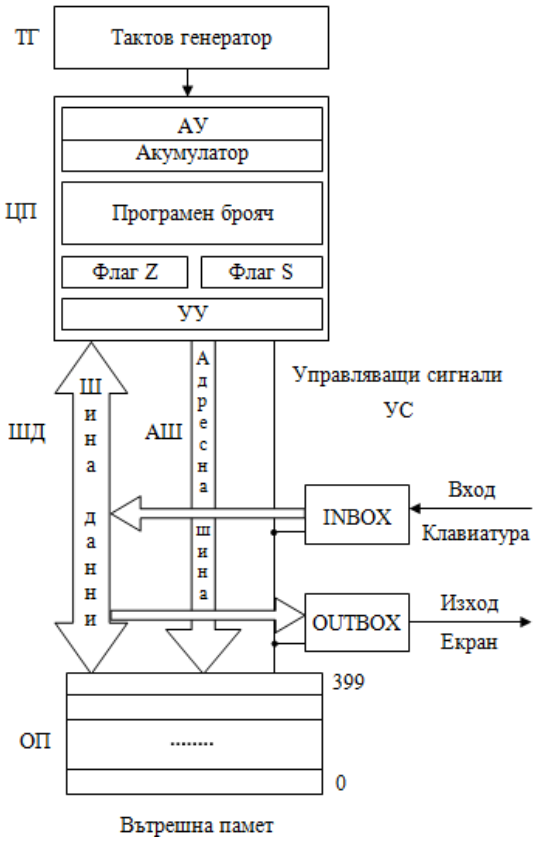
2. Архитектура на учебния микрокомпютър СЕКТ

С цел усвояване на основните архитектурни принципи на последователните микрокомпютри, е разработен модел на архитектурата на елементарен микрокомпютър с

типична фон-Нойманова архитектура, с название СЕКТ. Въпреки че е наречен елементарен, в структурата на микрокомпютъра са включени всички типични за реалните компютри устройства с фон-Нойманова архитектура – централен процесор, памет, входно/изходна система и обща магистрала, с някои ограничения (фигура 1). Симулираният микрокомпютър няма външна памет и система за прекъсване. В системата от 20 едноадресни инструкции има типичните за реални процесори групи от инструкции според предназначението им, с изключение на инструкции за обработка на прекъсванията и инструкции за изпълнение на логически операции. Инструкциите на микрокомпютъра са с размер 16 бита, но за разлика от 16- битовите микропроцесори, те са едноадресни. Двоичният код на инструкцията се разделя на две полета - поле от 4 бита за КОП, и поле от 12 бита за адреса на един операнд.

Основните регистри, които са програмно достъпни, са Акумулатора и Флаговия регистър. Акумулаторът е регистър за временно съхранение на единия от операндите и на резултата от изпълнението на указаната в инструкцията операция. АУ получава един от своите операнди от Акумулатора, резултатът от операциите винаги остава в него.

Програмният брояч съхранява адреса на поредната за извличане от паметта инструкция на програмата. Неговото съдържание автоматично се увеличава с 1 след всяка извлечена инструкция.



Фиг. 1. Архитектура на учебен микрокомпютър СЕКТ

Флаговият регистър в СЕКТ се състои само от два бита – **Z** (zero- нула) и **S** (sign- знак). Съдържанието на тези битове става равно на 1 става автоматично при следните условия: Z = 1, когато резултата от обработката на предишната инструкция в Акумулатора е нулев; S = 1, когато резулата в Акумулатора е отрицателен.

Входно - изходната система на симулатора е представена от две програмни „кутии”- INPUT BOX – Вход и OUTPUT BOX- Изход. Когато след стартиране на симулатора се изпълни инструкцията **read**, на екрана се изобразява поле с надпис INPUT BOX. В него потребителят може да запише стойност като входна величина. OUTPUT BOX се изобразява на екрана на симулиращия компютър, когато СЕКТ изпълни инструкция **print**.

Транслираните инструкции от асемблера се съхраняват в двоичен код във вътрешна памет, състояща се от 400 клетки с номера от 0 до 399. Всяка клетка от паметта може да съхрани една инструкция или число с плаваща запетая. Паметта на СЕКТ е „етикетна", с цел да не се извлича число като инструкция и да не се извършват аритметични операции над инструкции. Процесорът обменя данни само с клетки от паметта.

За разлика от други симулатори, в СЕКТ не се извършва обработка на прекъсванията на основната програма [1].

СЕКТ е софтуерен продукт, предназначен за начално изучаване на организацията и архитектурата на компютъра. Той симулира компютър с елементарна микроархитектура. Симулаторът включва софтуерна среда с команден интерфейс за симулиране на изпълнението на асемблерови програми под управлението на ОС ДОС (работи с всички версии на WINDOWS), компилатор на Асемблер, описание на системата от инструкции и описание на конфигурацията. Изходният код на програмата – над 5000 реда, е написан на езика „С".

Действието на СЕКТ се състои в последователното изпълнение на инструкциите от програмата. Цикълът на изпълнение на инструкцията се състои от 3 фази - извличане, декодиране и изпълнение. Изпълнението на всяка операция, съответстваща на инструкция, представлява последователност от изпълнения на няколко микрооперации. в синхрон със сигнала на тактовия генератор (ТГ).

3. Система от инструкции и програмиране на СЕКТ

Системата инструкции, с цел по-лесното им изучаване, се състои само от 20 инструкции, разделени на следните 3 групи, които са дадени с техните мнемонични кодове:

Инструкции за трансфер на данни:

load n – копира съдържанието на адрес **n** от паметта в Акумулатора;

loadc n – зарежда в Акумулатора константната величина **n**;

store n – съхранява съдържанието на Акумулатора в клетка от паметта с адрес **n** ;

read n – копира числото, от Input Box, в клетка от паметта, представена чрез **n**;

print n – изпраща копие на числото в клетка **n** от паметта, към изхода Output Box;

Аритметични инструкции:

add n – прибавя съдържанието на клетка с етикет **n** към това на Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете;

sub n – изважда съдържанието на клетка с име **n** от това на Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете;

mult n – умножава съдържанието на клетка с име N със съдържанието на Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете;

div n – дели съдържанието на клетка с име **n** със съдържанието на Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете;

addc n –прибавя константата съдържаща се в клетка с име **n** към съдържанието на Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете;

subc n – изважда константата съдържаща се в клетка с име **n** от съдържанието на Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете;

multc n – умножава константата **n** със съдържанието на Акумулатора, съхранява резултата в Акумулатора и установявайки състоянието на флаговете;

divc n – дели константата, съдържаща се в клетка с име **n** със съдържанието на Акумулатора, съхранява резултата в Акумулатора и установявайки състоянието на флаговете;

sqrt – изчислява квадратен корен от числото, съдържащо се в Акумулатора, съхранява резултата в акумулатора и установява състоянието на флаговете;

exp – изчислява резултата от повдигане на числото **e** на степен, равна на числото съдържащо се в Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете;

log – изчислява натурален логаритъм от числото, съдържащо се в Акумулатора, съхранява резултата в Акумулатора и установява състоянието на флаговете.

За разлика от други микропроцесорни системи, СЕКТ няма инструкции за изпълнение на логически операции от типа И, ИЛИ, НЕ и изключващо ИЛИ.

Инструкции за управление изпълнението на програмата:

jump n- безусловен преход, продължава изпълнението на програмата от инструкцията, в полето на която има етикет **n;**

jifz n- продължава изпълнението на програмата от инструкцията, етикетирана чрез **n**, ако флага Z=1, в противен случай изпълнява следващата непосредствена след нея инструкция ;

jifn n – продължава изпълнението на програмата от инструкцията, етикетирана чрез **n**, ако флага S=1, в противен случай изпълнява следващата инструкция;

stop – спира изпълнението на програмата;

Освен инструкциите, в асемблера на СЕКТ има една **директива** за инициализация на данни или памет.

n data – зарежда в клетка от паметта с име **n** стойност на величината с име **n**;

В симулатора се обработват двоични данни, но за улеснение при програмирането значенията на отделните величини и константите в програмите се записват в привичната за човека десетична бройна система.

Изходният код на потребителската програмата може да бъде написан чрез текстов редактор, или чрез възможностите на симулатора с комбинацията от клавиши **ALT + E**. Файлът на програмата с изходния код трябва да има разширение на файла **.txt.** Синтаксисът на всеки ред от програмата, има същите четири полета, както при Асемблера на Интел:

Етикет_ мнемоничен код на инструкция_ етикет или име на операнд_; коментар.

За деклариране и инициализация на променливите се използва директивата **data ,** в следния формат:

символично име на променливата (в поле етикет)_ data_ начална стойност

Изходен код на примерна програма за изчисляване на квадратен корен от израза **3a+2b,** при a=6 и b=5**,** с име на файла **3a2b.txt**:

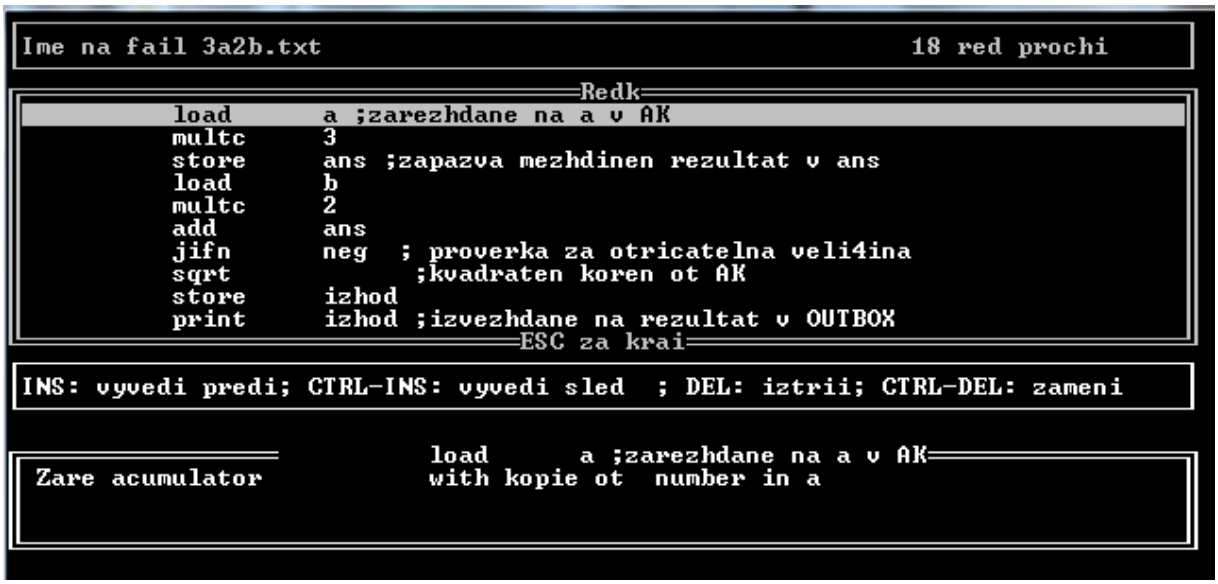
```
        load    a ;zarezhdane na a v AK
        multc   3
        store   ans ;zapazva mezhdinen rezultat v ans
        load    b
        multc   2
        add     ans
        sqrt        ;kvadraten koren ot AK
        store   izhod
        print   izhod ;izvezhdane na rezultat v OUTBOX
        stop
a       data    6
b       data    5
ans     data    0
izhod   data    0
```

Резултатът трябва да се изведе от Акумулатора в изходната кутия Out box. Тъй като процесорът е едноадресен, се налага запис на междинния резултат от Акумулатора в клетка с име **ans.** За да се отпечата крайния резултат в Акумулатора – **5.292** , той трябва да се

прехвърли най- напред в друга клетка от паметта - **izhod.** Това е елементарна програма, в която няма проверка на резултата в Акумулатора, дали в него не се е получило отрицателно число. За да се направи това, е необходимо с инструкция за условен преход **jifn** да се провери резултата в Акумулатора, преди да се изчисли квадратния корен.

След стартиране на операционната система, на екрана се появява прозорец за работа под DOS. Управлението на симулатора се извършва от екранен редактор чрез команди, активирани с комбинации от клавиши (например **ALT + F** зарежда файла на програмата). Предвидени са два режима на изпълнение на програмата – постъпково изпълнение на инструкциите „ трасиране" (комбинация от клавиши **ALT + R**) с възможност за редактиране, и изпълние на цялата програма (комбинация **ALT + G**). В режима „трасиране" обучаемият може да проследи в долната част на екрана съдържанието на програмния брояч, акумулатора и двата флага след изпълнението на поредната инструкция.

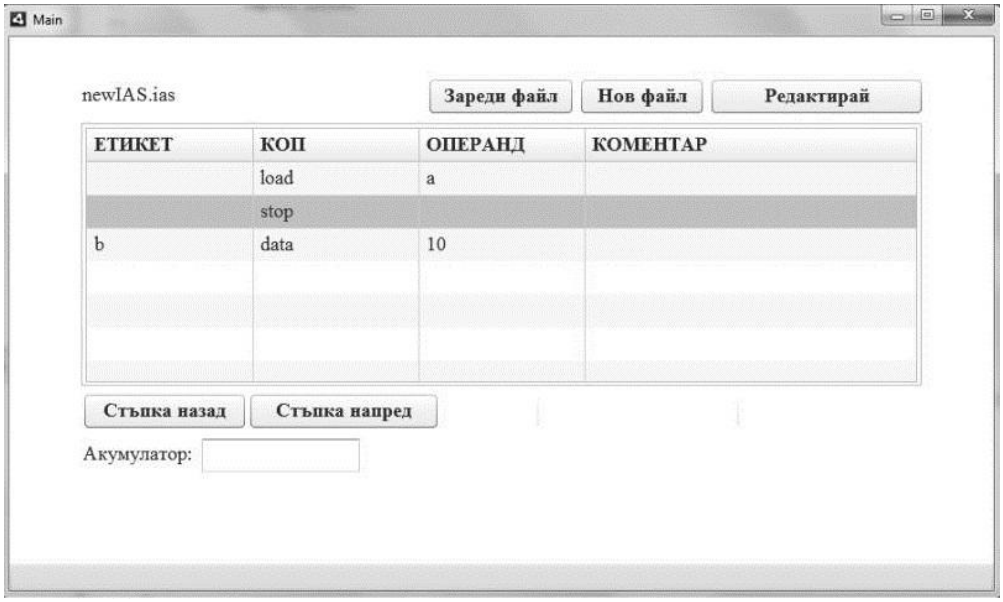
На фиг.2 е показан екран преди изпълнение от симулатора на първата инструкция от програмата с файл **3a2b.txt**. В полетата са дадени разяснения на български език, но с латински букви и съкращения на някои думи.



Фиг. 2. СЕКТ преди изпълнение на първата инструкция от програмата

4. Заключение

Опитът от използването във ФМИ на Шуменския Университет на разработения симулатор показва, че работата с него се усвоява много бързо от студентите, които нямат предварителна подготовка по компютърни архитектури. В перспектива работата продължава с неговото усъвършенствуване за подобряване на диалога с обучаемите чрез „дружелюбен" интерфейс (фиг.3) и въвеждане на нова система от инструкции, в която са включени инструкции за прекъсване на програмите **int** и **iret** (връщане след обработка на прекъсванията), за логически операции от типа **and n**, **or n**, и др. близки по абривиатура и функции с базовите инструкции на Intel – съвместими микропроцесори.



Фиг. 3. Обновен интерфейс на симулатора СЕКТ

Благодарности

Авторът благодари на Peter Smith от CMS - University of Greenwich, за оказаното съдействие при създаване на симулатора, на доц. Валентин Вичев за внедряването и тестването му в обучението във Факултет АПВОиКИС на НВУ'В.Левски", на гл.ас. Пламен Иванов от филиала на РУ „А. Кънчев" в гр. Силистра за направените ценни препоръки по усъвършенстване на симулатора, и на г-ца Валерия Георгиева, която извърши необходимите преработки на софтуера.

Литература

[1] TOM Computer Simulator. [онлайн].[прегледан 15.05.2009]. http://www.keylinkcomputers.co.uk/Tom/TOMSimulator/tomsimulator.htm.
[2] EMU8086 Microprocessor Emulator. [онлайн].[прегледан 15.06.2009]. http://www. emu8086. com/
[3] Станев, С., П. Смит и Ф. Иванов. Компютърни системи и мрежи. Шумен: Университетско издателство „Епископ Константин Преславски", 2002. ISBN 954-577-120-8.198 стр.

За контакти:
Доц. д-р инж.Станимир С. Станев
Катедра „КСТ"
Шуменски университет „Епископ Константин Преславски"
E-mail: stan@shu-bg.net

ПРИЛОЖЕНИЕ И ВЪЗМОЖНОСТИ НА СПЕЦИАЛИЗИРАНИЯ ЕЗИК ABEL ЗА ПРОЕКТИРАНЕ В XILINX CPLD XC9500

Горан Д. Горанов

Резюме: Програмирането и използването на различни езици и подходи е индивидуално за всеки програмист. Това зависи от неговия опит, ползвана елементна база, познаване на чужд опит и др. Най-често се изпозват близки по структура и логическа подреденост програмни средства. Разпространена възможност за проектиране в Xilinx XC9500 е **Advanced Boolean Expression Language** (**ABEL**), който включва едновременно логическите формати за уравнения и таблици на истинност, също и формат за частично описание на състоянието чрез граф. В допълнение към това, ABEL може да се използва и за описване на тестови вектори-шаблони за входове и очаквани изходи. Като структура езикът е близък до асемблер.
Ключови думи: VHDL, VERILOG, ABEL, JHDL, OPENVERA, CPLD, Xilinx .

Application and capability of ABEL language for designing in XILINX CPLD xc9500

Goran D. Goranov

Abstract: Programming and the use of different languages and approaches for each individual programmer. The fact is that you cannot know all possible languages, but can a designer to use and teach those who are close in structure and logical order. Advanced Boolean Expression Language (ABEL) includes both logical format for equations and truth tables also a state machine form - graph. In addition, ABEL can also be used to describe the test-vector patterns inputs and expected outputs. Its structure is similar to the language assembler.
Keywords: VHDL, VERILOG, ABEL, JHDL, OPENVERA, CPLD, Xilinx .

1. Увод

В областта на цифровата техника често се изпозват хардуерно ориентирани езици за програмиране (HDL), намиращи приложение при програмиране на цифрови вериги (цифрова логика). Описват се логическите операции на веригата, структурата и организирането й, както и има възможност за тестване и изследване чрез симулация.

HDL езиците са стандартни текстово–базирани изрази от пространствената и преходна структура на режима на работа на електронните системи. За разлика от повечето други софтуерни програмни езици, HDL езиците включват определена система за време, която е основна характеристика на хардуера. През 1983г. Data-I/O представя ABEL[6]. Той е разработен с цел да описва логически програмни устройства и е бил основно използван за структура на автомати с определени състояния. През 1987г., по молба от министерството на отбраната на САЩ се разработва VHDL (VHSIC Hardware Description Language), като VHSIC означава Very High Speed Integrated. За няколко години двата езика VHDL и Verilog се оказват доминиращи HDL езици за проектиране в областта на цифровата електроника, които постепенно изместват по-ранни версии на HDL. В същото време VHDL и Verilog притежават много ограничения[5,7]:

- Никой HDL не е подходящ за симулация на аналогова и аналого-цифрова верига.
- Никой не притежава езикови конструкции за програмиране на рекурсивно генерираните логически структури.

- Липсва възможността за оптимизация на логическите зависимости, което на практика създава предпоставка за използването на голям ресурс макроклетки или конфигурационни логически блокове.

Езикът ABEL има своите предимства въпреки доминиращите по използване VHDL и Verilog. Този HDL език вече две десетилетия се използва с успех от много програмисти на PLD структури. След серии постижения, ABEL сега се притежава от Xilinx Inc.

2. Изложение

ABEL-HDL е хардуерен език за програмиране, който поддържа входни променливи заедно с уравнения от високо ниво, граф на състоянията и таблици на истинност. ABEL компилаторът продуцира файлове за симулация и файлове за вграждане на структури в PLDs или FPGAs. ABEL-HDL проектирането позволява да бъде преобразуван в друг формат в зависимост от използваната програмируема среда.

Всеки ред в ABEL-HDL файла трябва да спазва следните синтактични правила[4]:

- На един ред се побират до 150 символа;
- Редът завършва чрез добавянето на нов ред (hex 0A), чрез вертикално разделяне (hex 0B), чрез нов формат (hex 0C);
- Думите и цифрите са разделени с единично място. Изключения от това правило са списъци с идентификатори, разделени със запетаи, където изрази идентификатори или числата са разделени от операторите;
- Може да се оставя пространство за коментари, блокове и фактическите обстоятелства. Например, ако ключовата дума MODULE се въведе като MOD_ULE, тя се разглежда като два идентификатора, MOD и ULE;
- Идентификаторите като имена и етикети може да са с главни букви или малки букви, но са зависими от това (keys sensitive);
- Всички големи и малки букви и други синволи са валидни и се използват. Приложими са следните: a - z (малки букви), A - Z (големи букви), 0 - 9 (цифри), <space}, <tab>, ! @ # $? + & * ( ) - = + [ ] { } ; : ' " ` \ | , < > . / ^ %.

Таблица 1 – команди в ABEL

async_reset,	library
case,	macro
declarations,	module
device,	node
else,	option
enable (obsolete)	pin
end	property
endcase	state
endwith	state_diagram
equations	state_register
external	sync_reset
flag (obsolete)	test_vectors
functional_block	then
fuses	title
goto	trace
if	truth_table
in	when
interface	with
istype	

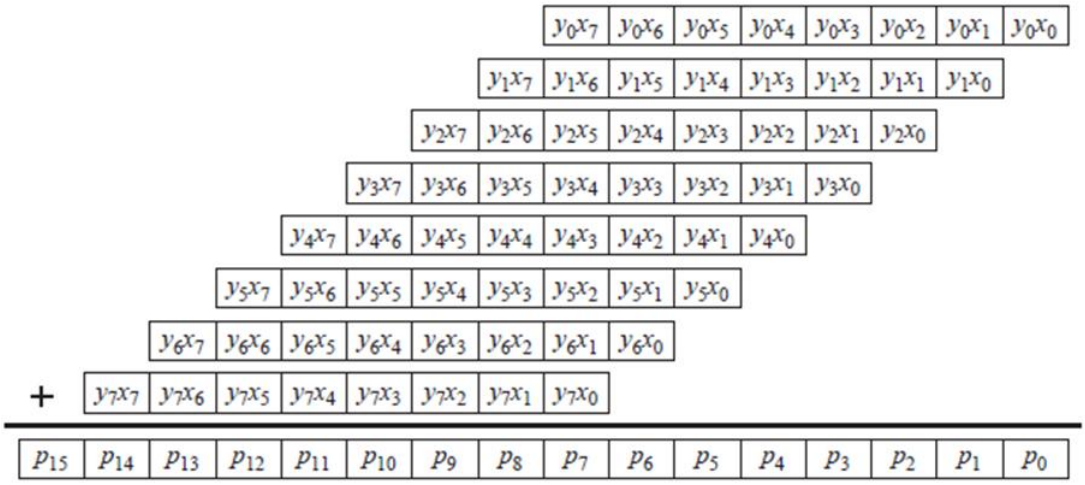
ОПЕРАТОР	ПРИМЕР	ОПИСНИЕ
=	A = 5	combinatorial assignment
:=	A := [1,0]	registered assignment
?=	A ?= [1,0]	combinatorial don't care assignment
?:=	A ?:= 5	registered don't care assignment
!	!A	NOT: ones complement
&	A & B	AND
#	A # B	OR
$	A $ B	XOR: exclusive OR
!$	A !$ B	XNOR: exclusive NOR
-	-A	negate
-	A - B	subtraction
+	A + B	addition
==	A == B	equal
!=	A != B	not equal
<	A < B	less than
<=	A <= B	less than or equal
>	A > B	greater than
>=	A >= B	greater than or equal

Фиг. 1. Описание на операторите

Съществуват специални думи (команди), които са запазени за идентификатори. Те не могат да се използват като имена на модули, изводи, константи и макроси (таблица1).

Използваните оператори са разделени на категории съответно логически, математически, присвояващи и изразяващи връзка. Примери за използването на операторите и пояснение към тях са дадени на фигура 1. По-долу е посочен пример – умножение на две двоични числа.

Умножение на две двоични числа може да се реализира чрез преместване на ляво толкова пъти, колкото разреда е числото. Алгоритъмът е показан на фигура 2, който е познат и използван в контролери, които нямат хардуерно умножение.



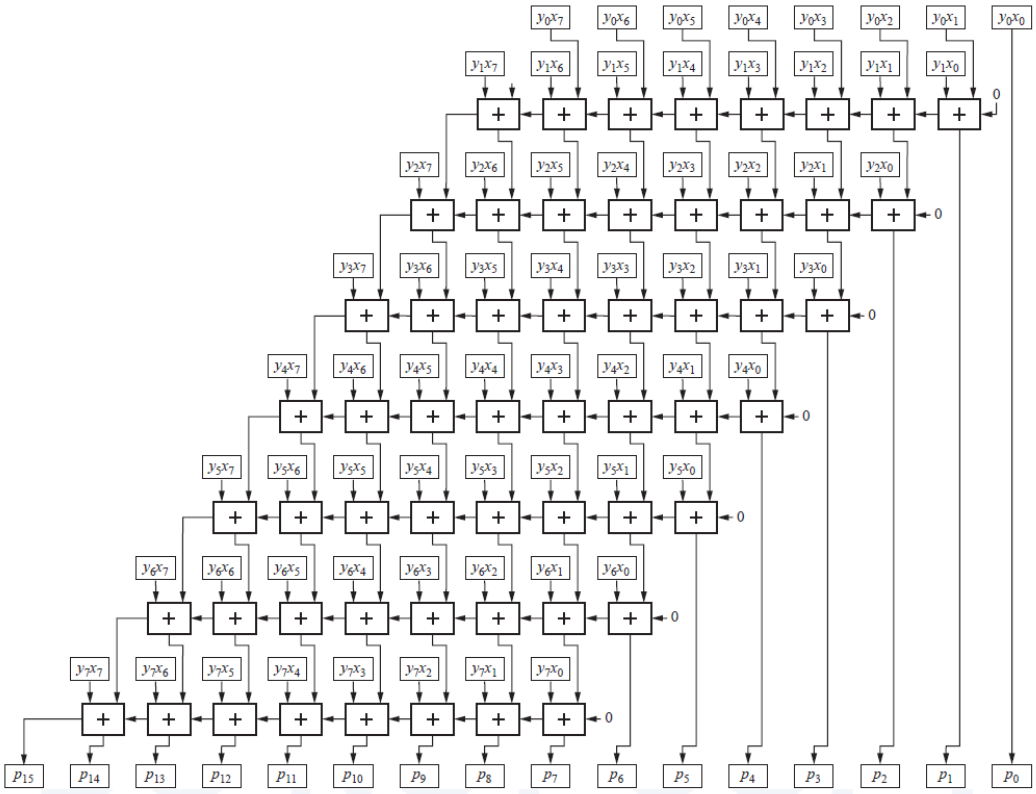
Фиг. 2. Алгоритъм за умножение на две 8 битови числа

Прави се побитово умножение на две 8 битови числа $X = x7x6x5x4x3x2x1x0$ и $Y = y7y6y5y4y3y2y1y0$, като в последствие частичният резултат се събира отново по битове за получаване на 16 разреден резултат, т.е. произведението е P=X.Y. Тази функция е комбинационна, което значи, че може да се реализира с логически елементи.

За сравнение в един микроконтролер реализацията на тази функция се изпълнява от АЛУ по алгоритъм, показан на фигура 3.

Представеният блоково алгоритъм би могъл да се реализира чрез използването на логически елементи И – ИЛИ, но неговата сложност ще изисква ресурс и време. Разработената програма за умножение, показана по-долу, дава представа как с по-малко действия и време може да се реализира умножител на две N- битови числа. Примерът е за 4 битови числа, но с промяна на кода се постига умножение на примерно 8 битови числа или N битови.

```
Module mul4x4
title '4x4 Combinational Multiplier'
X3..X0, Y3..Y0 pin;
P7..P0 pin istype 'com';
P = [P7..P0];
PC1 = Y0 & [0, 0, 0, 0,X3,X2,X1,X0];
PC2 = Y1 & [0, 0, 0,X3,X2,X1,X0, 0];
PC3 = Y2 & [0, 0,X3,X2,X1,X0, 0, 0];
PC4 = Y3 & [0,X3,X2,X1,X0, 0, 0, 0];
equations
P = PC1 + PC2 + PC3 + PC4;
End mul4x4
```



Фиг. 3. Алгоритъм за умножение в микроконтролер

Заключение

Чрез настоящата разработка са прставени възможностите и гъвкавостта на ABEL. Изпозван е софтуерен продукт на фирмата XILINX [1,2], като посоченият умножител и много други цифрови модули (комбинационни схеми, комбинационни блокове, брячи, и др.) са реализирани в CPLD XC95108PC84-15. С такава структура този език е по-близък и удобен за изучаване от студенти и проектанти на микроконтролерни системи с асемблер.

Литература

[1]. Foundation Series 2.1i User Guide, Chapter 5, Design Methodologies - HDL Flow, available on the Xilinx website.

[2]. D. Van den Bout, "The Practical Xilinx Designers Lab Book 1.5", Prentice Hall, Upper Saddle River, 1999.

[3]. J. Wakerly, "Digital Design", 2nd Edition, Prentice Hall, Upper Saddle River, 2000.

[4]. ABEL design manual - Lattice Semiconductor Corporation, 5555 NE Moore Ct. Hillsboro, OR 97124 (503) 268-8000, March 2003.

[5]. Using Verilog to Create CPLD Designs XILINX XApp 143 2001.

[6]. J. Mermet (editor): Fundamentals and Standards in Hardware Description Languages (Springer Verlag, 1993)

[7]. Barbacci, M., Grout S., Lindstrom, G., Maloney, M.P. "Ada as a hardware description language: an initial report," Carnegie-Mellon Univ., Dept. of Computer Science, 1984

За контакти:

Гл. ас. д-р инж. Горан Д. Горанов

катедра „Електроника”

ТУ-Габрово

E-mail: g_goranov@bitex.bg

СИСТЕМА ЗА УПРАВЛЕНИЕ НА СЪДЪРЖАНИЕТО

Горан Д. Горанов, Искрен Кандов

Резюме: С развитието на интернет технологиите, както и с увеличаването броя на уеб базираните приложения, нарастват и изискванията към тях. При наличието на толкова голям брой интернет сайтове, стремежът към интерактивност и разнообразие нараства. Чрез система за управление на съдържанието могат да се променят модулите, изграждащи уеб сайта, а именно статии, галерии и други.
Ключови думи: CMS, PHP, MySQL, LMS

Content management system

Goran Goranov, Iskren Kandov

Abstract: With the development of the Internet technology and the increasing number of web-based applications the applications' requirements also increase. The availability of too many websites leads to increase of the interactivity and diversity requirements. The CMS (Content Management System) helps management of the site and its interactivity as web application. The CSM system provides functionality for control of the content of the site by controlling site's modules as galleries, news and etc.
Keywords: CMS, PHP, MySQL, LMS.

1. Увод

Системата за управление на съдържанието представлява съвкупност от скриптови файлове и бази данни. Познатите такива системи с отворен код са реализирани на базата на .php файлове и работа с MySQL бази данни. При динамичните уеб сайтове работата с бази данни е задължителна, като основно служат за съхранение на информация. Добавяне на информация, както и нейното извличане става посредством скриптове, а времето, необходимо за тези операции, е минимално. Сам по себе си PHP (Hypertext Preprocessor) представлява скриптов език, който се изпълнява на сървъра, и не се нуждае от компилация. Предимството на езика се състои в това, че не е необходимо да се дефинира вида на променливите [1, 2].

Системите за управление на съдържанието улесняват значително създаването, обновяването или изтриването на информация в уеб сайта. Разработването на такава система обаче, изисква не само познания по скриптови езици, като PHP, Python, Perl и други, но и владеене на SQL за моделиране и извличане на данните от релационна база от данни. В случаите, когато липсват познания по SQL и уеб сайтът е изцяло концентриран върху управление (показване, въвеждане и обновяване) на данните от базата, са известни и други решения. Примерни такива са разгледани от авторите в [3].

Системите за управление на съдържанието могат да се разделят на два основни типа:

- Индивидуални системи за управление (системи, които се разработват за конкретен уеб сайт);
- Универсални системи за управление (JOOMLA!, Word Press, Drupal).

Универсални могат да се нарекат тези системи, които могат да бъдат имплементирани във всеки уеб сайт. Те съдържат в себе си голям брой модули за управление на съдържанието. Тяхната универсалност се базира на работата с тези модули, включително тяхното конфигуриране [1]. В голямата си част тези системи са безплатни - т.н. отворен код. Използват се от голям брой потребители в света, което спомага за развитието на системите.

Една от най-използваните системи за управление на съдържанието е JOOMLA!

Joomla! е безплатна система за управление на съдържанието с отворен код, написана изцяло на PHP и използваща MySQL бази данни. Тази система се използва навсякъде по света за изграждане на прости лични страници до комплексни уеб приложения. Тя може да бъде използвана за лесно управление на уеб сайта, от добавяне на статии и снимки до обновяване на продуктов каталог или приемане на онлайн резервации [7].

Управлението на сайта се извършва чрез опростен интерфейс, който позволява лесното публикуване на новини, обяви за работа, продукти, да се създават секции, категории и менюта. Администраторите на уеб сайтове, които са изградени на базата на JOOMLA!, не е необходимо да са запознати с работата на сложните скриптове, изграждащи системата, както и с работата с бази данни. Това е голямо предимство, което се използва от много потребители за изграждане на уеб приложения.

Друга подобна система за управление на съдържанието е Drupal [7, 8]. Drupal е модулен софтуерен пакет с отворен код, който дава възможност на определена общност от потребители да публикуват, да управляват лесно разнообразното съдържание на уеб страница. Drupal може да бъде използвана като система за управление на съдържание, но притежава свойства и на таксономична система. Изградена е на базата на PHP и работи с MySQL бази данни [1, 2, 7, 8].

Ядрото на Drupal се състои от помощни елементи за по-лесната и бърза изработка на регистрации, менюта, RSS и уеб администриране. В същността си определен сайт, изработен чрез Drupal, представлява множество от блогове, обединени в един общ блог. Както при JOOMLA!, така и Drupal се използва безплатно, унифицирана относно основните изисквания към уеб базираните приложения, предлага удобна интерактивна за управление визия с множество опции [7].

И двете системи работят успешно по отношение на управление на уеб съдържание. Модулите, които могат да се интегрират във всяка система, нарастват всеки ден, което разширява и техните възможности. Основното предимство е и основният недостатък на тези системи, а именно тяхната универсалност. Те се изграждат да задоволяват някои от най-използваните потребности при администриране на уеб приложения, но колкото и универсални да са, не могат винаги да отговарят на 100% на функционалността на всеки сайт. За да може един сайт да бъде напълно функционален, както и да може да се разширява и да бъде уникален, се изработва индивидуална система за управление.

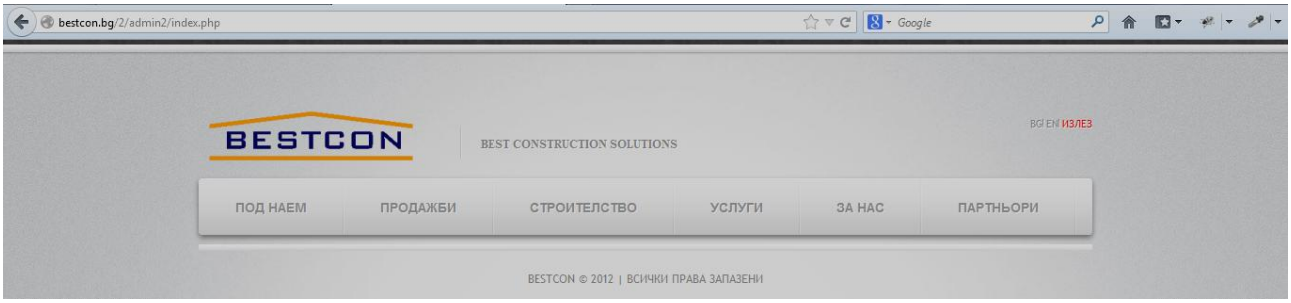
Целта на настоящата статия е да се разработи индивидуална система за управление на уеб сайт, както и да се сравни с досега познатите универсални такива.

2. Индивидуална система за управление на съдържанието

Разработена е система за управление на съдържанието, която се базира на работа с PHP 5.0, както и на използването на бази данни от типа MySQL. За пълен достъп в системата е необходимо да се въведе съответният адрес, както и легитимирането с потребителско име и парола. За допълнителна защита се използва и криптиране на данните чрез използване на MD5 кодиране [1, 2, 4, 5].

Системата за управление притежава удобен и лесен за използване интерфейс – фигура 1, с помощта на който лесно могат да се добавят:

- Категории;
- Подкатегории;
- Изображения;
- Текстова информация;
- Файлове и др.

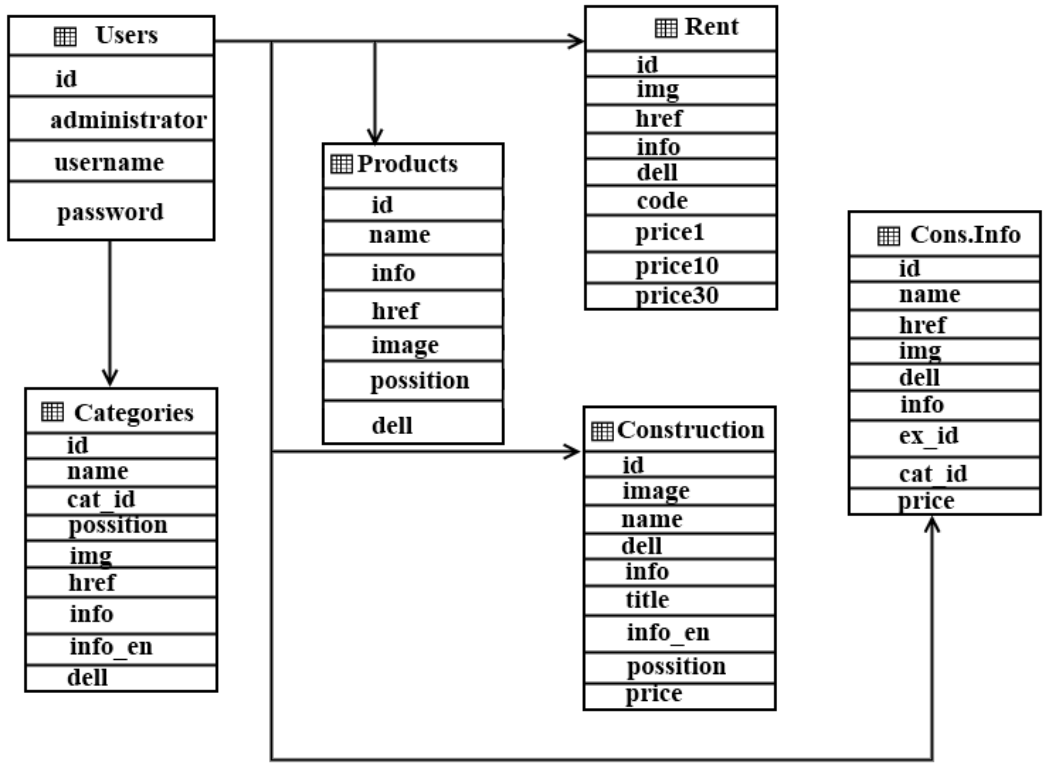


Фиг. 1. Основен изглед на системата за управление

Системата е внедрена в уеб сайт за отдаване под наем на строителна техника, както и нейната продажба. За да отговаря на заданието, е необходимо разширяване на функционалните възможности на системата за управление. Допълнително е разработен скрипт, чрез който може да се добавят шифри на всеки продукт, цена за 1 ден, цена за 10 дни, както и цена за 1 месец. [1, 2, 5]

За да може информацията да се съхранява, редактира, изтрива и обновява, тя трябва да бъде запазена някъде, където да е лесно достъпна. За целта се използват бази данни, които служат за съхраняването на информацията на сайта.

Структурата на базата е показана на фигура 2. Вижда се, че базите данни са така създадени и структурирани, че администраторът да може да управлява всяка таблица поотделно.



Фиг. 2. Структура на базите данни.

Във всеки един момент от време администраторът може да следи съдържанието на всяка от таблиците, както и може да редактира и изтрива съдържанието им [1, 2, 4].

Администраторът може да реализира следните операции чрез използване на системата за управление:

- Създаване на нови продукти;
- Добавяне на услуги;
- Определяне на цени;
- Добавяне на информация(изображения, текст и др.);
- Редактиране, изтриване и подредба на досега реализираните продукти;
- Добавяне на различни модули (брояч на посетители, електронен магазин и др.).

Основно предимство на уеб базираните системи за управление е това, че могат да се ползват отвсякъде, където има достъп до глобалната мрежа. Разработената индивидуална система за управление е приложима само за този уеб сайт. Това гарантира висока сигурност и защита на информацията от зловредни намеси.

Недостатък на универсалните системи е свързването на базата от данни със съответните модули. То може да не е оптимизирано, а това води до намаляване на функционалността на сайта. При индивидуалната система за управление модулите са разработени да отговарят напълно на функционалността на уеб приложението.

Заключение

Разработената система за управление повишава функционалността на уеб базираното приложение, за което е разработена. Намаляване на времето за комуникация е постигнато с оптимизация на базата от данни. Този подход за разработка на индивидуални системи за управление на съдържанието намира все по голяма популярност, особено при сложни и нестандартни уеб сайтове. Разработената система е внедрена успешно в сайт за отдаване на строителна техника на фирмата БЕСТКОН, намиращ се на адрес: www.bestcon.bg/2.

Литература

[1] Нарамор Е., Джейсън Г., Програмиране и Web дизайн с PHP5, MySqL, Apache том първи – 2005г.
[2] Нарамор Е., Джейсън Г., Програмиране и Web дизайн с PHP5, MySqL, Apache том втори – 2005г.
[3] Goranova R. D., Armyanov, P., GUI Tools for complex queries in RDBMS, Conference Proceedings of the 5th International Conference ISGT, 27 – 28 May 2011, Sofia, Bulgaria, pp. 75-81.
[4] http://www.w3schools.com/php/default.asp. -2013г.
[5] http://www.w3schools.com/html/default.asp - 2013г.
[6] http://www.mysqltutorial.org/ -2013г.
[7] http://www.joomla.org/ - 2013г.
[8] http://drupal.org/ - 2013г.

За контакти:
Гл. ас. д-р инж. Горан Д. Горанов
инж. Искрен Кандов
катедра „Електроника”
ТУ Габрово

E-mail: g_goranov@bitex.bg
E-mail: i.kandov@mail.bg

GENERATION OF STEREO IMAGES IN 3D GRAPHICS APPLICATIONS FOR STEREOSCOPIC AND NONSTEREOSCOPIC DISPLAYS

Emiliyan G. Petkov

Abstract: This article discusses generation of stereo images by 3D graphics applications. Having stereo images and appropriate device for observation the viewer receives two different images for both eyes. Then the brain makes the depth. Such depth perception can be useful in many fields, for example, scientific visualization, entertainment, games, virtual reality applications, validation of architectural objects, etc. For correct generation of stereo images the work of the human eyes should be taken into consideration. The paper gives different approaches for generating stereo images using two cameras in 3D computer generated virtual worlds. The article demonstrates the anaglyph technology – the easiest way to have stereo images on every personal computer.
Keywords: stereo, image, stereoscopy, 3D, display, graphics, application, anaglyph.

1. Introduction

Creation of stereo images is the technology dealing with two-dimensional drawings or photographs that when one observes by both eyes he/she perceives the objects in three dimensions (in space, not just in a flat plane). The technique and science dealing with it is called Stereoscopy [2]. The pictures are produced in pairs (Figure 1). The members of a pair show the same scene or object from slightly different angles that correspond to the angles of vision of the two eyes of a person looking at the object itself. Stereoscopy is possible only because of human binocular vision, which requires that the left-eye view and the right-eye view of an object be perceived from different angles [6]. In the brain the separate perceptions of the eyes are combined and interpreted in terms of depth.

Figure 1. A stereoscopic image watched through a pair of glasses

A stereo view can make a picture much more intelligible. When the picture is viewed properly the viewer obtains a sense of depth in it. Even one stereo image (a pair of images) creates a virtual three dimensional image in front of the observer. A stereo view can makes the picture much more interesting and realistic and also reduces the visual ambiguity in it (such is which lines lay in front of others) as well.

Stereoscopy is only one of the major three dimensional display technologies. There are technologies that are autostereoscopic, such as holographic systems, lenticular (barrier strip) systems and one-view or multiview autostereoscopic 3D displays [7,8]. Some stereoscopy systems require additional devices such as special monitors, polarizing glasses and the high-end graphic

cards. These are special systems and devices which are very expensive. This article also discuses a stereoscopic technology that is applicable on every personal computer.

Stereoscopic pictures are viewed when the right eye can see only the right-eye image and the left eye can see only the left-eye image. Special glasses are used to allow each eye to see only the appropriate picture of the pair. There are different technologies that could be viewed in [7]. Some stereoscopic systems are autostereoscopic which means that glasses are not needed.

There are different non-computer technologies for creating and watching stereo images. But techniques on how developers to create stereo images in their graphics applications and watch them on computer displays are required. This research aims to give an approach for stereo images generation in 3D graphics applications.

2. Perceiving the three dimensional world

2.1. Human eyes

Human ability to perceive the world in three dimensions (3D) and the distance of an object is called perception of depth [5,6]. Depth perception arises from a variety of depth cues. These are typically classified into binocular cues that require input from both eyes (Figure 2). Binocular cues include stereopsis, yielding depth from binocular vision through exploitation of parallax.

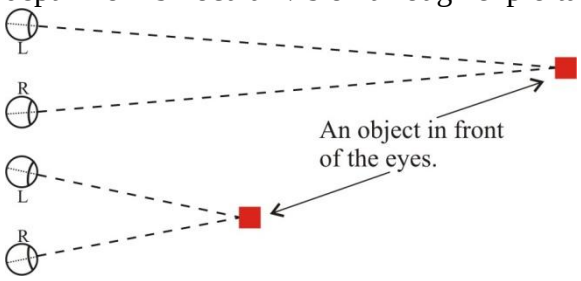


Figure 2. Human eyes convergence

The two human eyes receive two different images. This is because they converge to point to the same object but they have distance between each other (about 6.5 cm). This causes an object to be viewed from two different angles. The two images are sent to the brain and it gives us the perception of depth.

The approximate field of view of a human eye is 95° out, 75° down, 60° in, 60° up. About 12–15° temporal and 1.5° below the horizontal is the optic nerve or blind spot which is roughly 7.5° high and 5.5° wide.

Human eyes have horizontal displacement. The distance between the left and right eye projections is called the horizontal parallax of the eyes.

2.2. Cameras in 3D applications

Every 3D graphics system contains a type of an object called camera (Figure 3) [4]. The cameras in 3D graphics applications are used to generate images from the 3D scene (Figure 3). They simulate the work of real cameras. They can realize different types of projections. The projection that is mostly used and gives a correct view of the third dimension is perspective. The obtained images have dimensions in pixels.

When a camera is defined then position in the space, orientation and a shape of the viewing frustum (volume) should be determined. Concerning orientation every camera is defined by *u*, *v* and *w* directions; *v* is the direction of view, *w* points up and *u* points to the right from the camera [1].

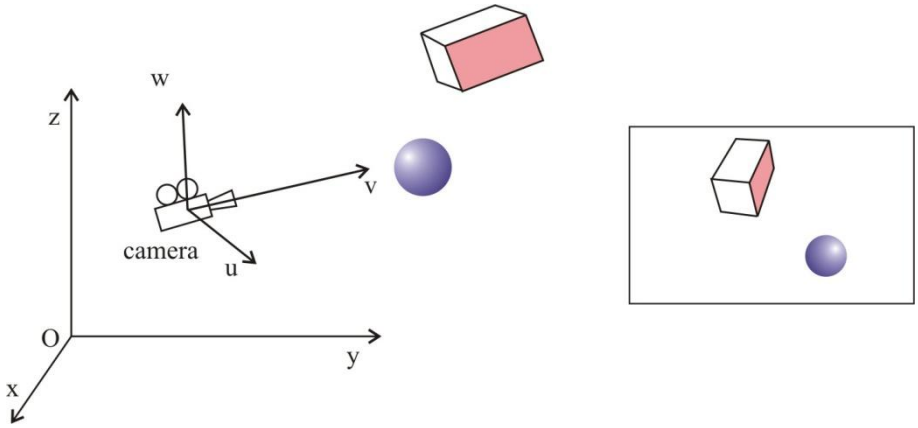


Figure 3. A camera and objects in a 3D space and the image from the camera

Also the shape of the viewing frustum should be determined. A near plane and a far pane must be fixed. This determines the field of view along *v*. Because a rectangular image is needed, the shape of the viewing volume is a rectangular pyramid, as given in Figure 4. So, the four angles of every side (face) of the pyramid in relation to *v* axis must be set. For example, if the image should have an aspect ratio 4:3, then the following angles could be set: 30° left, 30° right, 22.5° bottom and 22.5° up. Then the camera has a horizontal aperture θ of 60° and a vertical aperture γ of 45°. Also a camera could be set to simulate the work of a human eye. In this case the viewing volume of the camera should have values close or equal to the viewing angles of an eye.

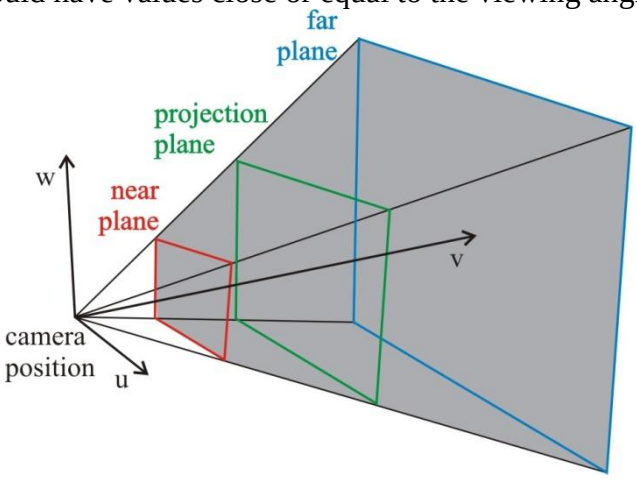


Figure 4. Camera view frustum

3. Obtaining stereoscopic images

The task of this research is human stereoscopic viewing to be simulated in 3D graphics applications. Humans have two eyes, so two images are needed to perceive depth. The decision is generation of stereo images. Two base methods will be discussed in regards to the placement of both cameras. First uses a convergence of the *v* axes of the cameras in a point in-front of them and second sets the *v* axes to be parallel. Both of them use cameras which work in perspective projection.

3.1. Convergence technique

Following [4], to obtain two pictures, a left and right, slightly different cameras are used, as suggested in Figure 5. The two cameras are built using the same look-at point but different positions. But where to be put the left and the right cameras must be decided. In [4] a simple

approach is given. It begins with a regular camera (named Cyclops) based on a single look-at point and a single initial Cyclops eye (Figure 5). A vector up is chosen. These establish the Cyclops camera with its *u*, *v* and *w* directions.

The left and right cameras are defined at slight displacements of the Cyclops eye, at some distance D along –*u* and *u*, respectively. The choice of D depends on the unit of measure being used in the application.

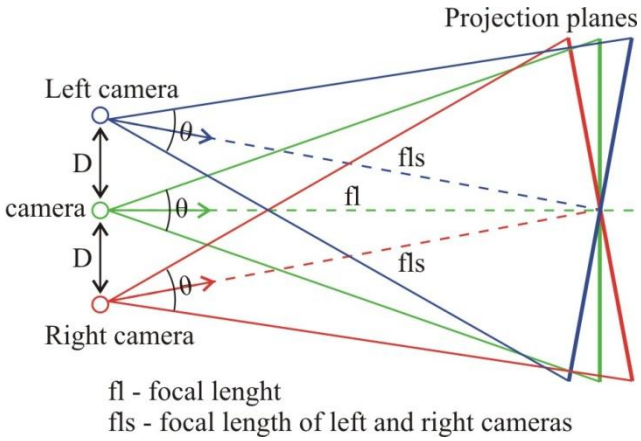


Figure 5. Left and right, slightly different cameras are set (top view)

In this camera arrangement *v* axes of the cameras are set to verge at a point in the scene. Objects in the same plane where the convergence point lays appear in the image plane of the final display if no other adjustments are made. Other objects can appear in-front of or behind the image plane of the display depending whether they were in-front or behind the convergence point. Objects that lie in front of the projection plane will appear to be in front of the display and objects that are behind the projection plane will appear to be into the screen.

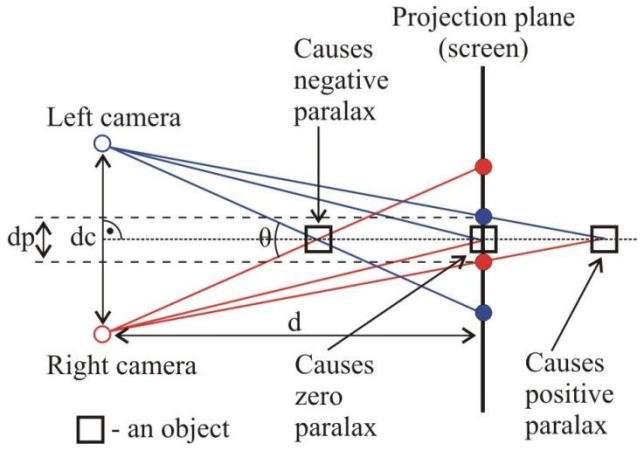


Figure 6. Positive, zero and negative parallax (top view)

The amount of image disparity, which is expressed in 3D depth, depends primarily on the camera separation, but is also directly affected by the cameras zoom settings, the focal length and the object sizes and distances from the convergence point [1,9].

Since the convergence point is behind the projection plane, it is called a positive parallax. Look at Figure 6. Since the convergence point lies to the projection plane, it causes a zero parallax. And since an object is in-front of the projection plane, it is known as a negative parallax. In this case the projection for the left eye is on the right and the projection for the right eye is on the left. As the object moves closer to the viewer, more than half way between the projection plane and the cameras, the negative horizontal parallax increases to infinity.

The disadvantage of the method is designated by the cameras in Figure 5. The projection plane of each camera is shown. The projection planes are not parallel and they are rotated in

opposite directions. This results to distortions in the left and right images – in particular different height distortions that introduce artificial vertical disparities which can become uncomfortable to view.

The disadvantages of this method could be accepted for big space observation. In this case convergence of the camera axes is the better way to ensure the whole field of view of the camera. But when objects which are close to the cameras should be taken this method is not recommended.

3.2. Parallel techniques

The convergence technique has disadvantages that can not be ignored. They come from the convergence of the cameras. The solution is cameras viewing directions to be set parallel. Creation of stereo images involves rendering of left and right camera views from two positions separated by a chosen distance. Both cameras work in perspective projection. The camera could look along parallel vectors. Then the frustum from each camera to the projection plane is symmetric. In this case a part of the left image does not exist in the right image and vice versa. But, after rendering, those parts of the images could be trimmed off. The system of two stereo parallel cameras with symmetric frustums is illustrated in Figure 7.

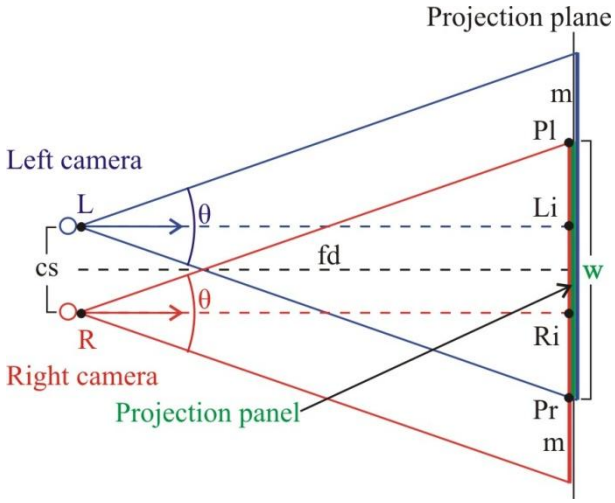


Figure 7. Stereo-camera system with symmetric frustums (top view)

In this case the degree of the stereo effect depends on both the distance *fd* of the cameras to the projection plane and the separation *cs* of the left and right cameras. Good results are achieved when not too large separation is set. Otherwise a hyper-effect is observed. A good stereo-camera system, for example, may use a horizontal aperture θ of 60° and a vertical aperture γ of 45°, a separation of the two cameras 1/25 of the distance to the projection plane.

In graphics application usually an image with appropriate dimensions is wanted. In case of a stereo-camera two images (left and right) with equal dimensions are needed. Let these dimensions be *w* (width) and *h* (height) and cameras be created in a graphics application. After this the focal distance *fd* and the camera separation *cs* should be set. Setting an appropriate value of *cs* parameter can be done in the particular graphics application. A value of 1/25 of the focal distance gives very good result (Figure 11). Now the horizontal aperture θ should be calculated, having in mind that $\theta \in (0, \pi)$. Both frustums are identical. So, we could look only at the left one (Figure 7). We have:

$|LLi| = fd$, $|Li\,\mathrm{Pr}| = \frac{w}{2} + \frac{cs}{2}$, $\angle LLi\,\mathrm{Pr} = 90°$, $\angle LiL\,\mathrm{Pr} = \frac{\theta}{2}$ and $\frac{\theta}{2} \in \left(0, \frac{\pi}{2}\right)$, then

$$tg\frac{\theta}{2}=\frac{\frac{w}{2}+\frac{cs}{2}}{fd} \Rightarrow \theta = 2arctg\frac{w+cs}{2\,fd}. \quad (1)$$

Analogous the value of the vertical aperture γ could be calculated:

$$tg\frac{\gamma}{2}=\frac{h}{fd} \Rightarrow \gamma = 2arctg\frac{h}{fd}. \quad (2)$$

Finally, the dimensions (*W* and *H*) of the images that both cameras are expected to generate before trimming should be determined. Both generated images by the render and trimmed off images have a same height *h*; and generated images by the cameras should be extended with *m* along width (but *m=cs*). So, the dimensions of the images that both cameras are expected to generate will be:

$$W = w+cs, \quad H = h. \quad (3)$$

To combine the images, *m* is trimmed off the left of the left image and *m* pixels are trimmed off the right of the right image.

Let's look at an example. If we want to set to a stereo-camera system the following values:

$fd = 700$, $cs = \frac{1}{25}.fd = 28$, $w = 800$ and $h = 600$,

then for the horizontal and vertical apertures of the cameras, and the width and height of both images we receive respectively:

$\theta = 61.2°$, $\gamma = 46.4°$, $W = 828$ and $H = 600$.

To create the stereo image we should trim off the left image along height and to take a sub-image with *x* values of the pixels in the following range [28, 827]; and the right image along height and to take a sub-image with *x* values of the pixels in the following range [0, 799].

May be the biggest disadvantage using this method is the need of trimming. But it has valuable advantages and two of them are: no vertical parallax and the frustums of both cameras are symmetric. To avoid the main disadvantage of this method a parallel technique with asymmetric frustums will be presented.

Let the dimensions of every one of both images be *w* and *h* and two cameras be created in a graphics application. Then the *v* axes of both cameras are set to be parallel and the focal distance *fd* and the camera separation *cs* should be set as well. After this the projection plane should be determined. Finally the apertures of both cameras should be calculated. Setting the frustums of the cameras we want one projection panel to be achieved for both cameras. Other words, having the frustums in Figure 7, we extend them as follows: left one to the right end of the projection panel (to point Pr) and right one to the left end of the projection panel (to point Pl). So, we receive asymmetric frustums. Let mark the horizontal aperture with θ and vertical one with γ, having in mind that $\theta, \gamma \in (0, \pi)$. Let's look at Figure 8. $\angle LPlPr$ and $\angle RRiPr$ are identical. So, we could discuss only over the left camera frustum. Let $\theta 1 = \angle LiLPl$ and $\theta 2 = \angle LiLPr$, so $\theta = \theta 1 + \theta 2$. Now we have:

$|LLi| = fd$, $|LiPl| = \frac{w}{2} - \frac{cs}{2}$, $\angle LLiPl = 90°$ and $\theta 1 \in \left(0, \frac{\pi}{2}\right)$, then

$$tg\theta 1 = \frac{w-cs}{2.fd} \Rightarrow \theta 1 = arctg\frac{w-cs}{2\,fd}; \quad (4)$$

$$|LiPr| = \frac{w}{2} + \frac{cs}{2}, \ \angle LLiPr = 90°, \text{ then } tg\theta 2 = \frac{w+cs}{2.fd} \Rightarrow \theta 2 = arctg\frac{w+cs}{2\,fd}. \quad (5)$$

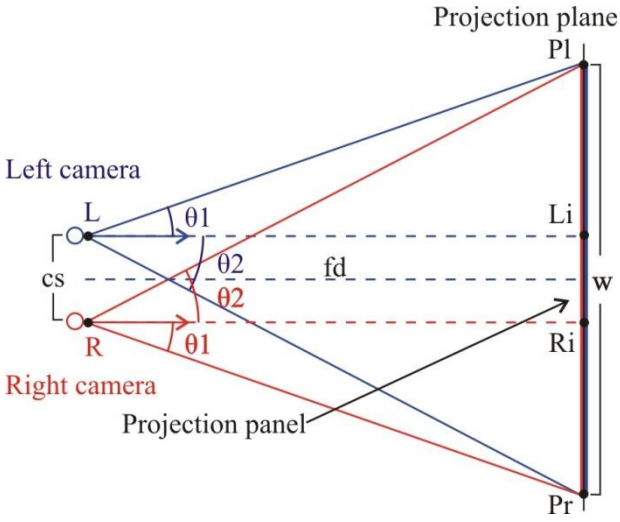


Figure 8. Creation of asymmetric frustums

Analogous the value of the vertical aperture γ could be calculated:

$$tg\frac{\gamma}{2} = \frac{h}{fd} \Rightarrow \gamma = 2arctg\frac{h}{fd}. \quad (6)$$

There is no photographic camera that has a possibility to take pictures using asymmetric aperture and frustum. But there are graphics systems that have a capability to do this [4]. This is because they work with perspective projection and it allows asymmetric frustums.

Advantages of this method are both: no vertical parallax and no trimming.

4. 3D anaglyph graphics application

Anaglyph images are used to provide a stereoscopic 3D effect, when viewed with glasses where the two lenses are different colors, such as red and cyan. The picture contains two differently filtered colored images, one for each eye. These two images may be produced by a stereo-camera system. Images are made up of two color layers, superimposed, becoming one image. With the two lenses of different colors every one of the eyes can see its own image. Thus a depth effect is achieved [4,6].

Anaglyph applications we may call computer applications that produce anaglyph images on the screen. These images could be static or movable. This means that movements on the screen of the computer are expected. The main advantage of anaglyphs is that they can be viewed with a minimum of hardware and expense. This kind of glasses is cheap enough for anyone and they are distributed even in magazines.

The glasses have two different color filters – one for each eye, for example, red on the right eye and blue on the left eye. The image that is destined for the left eye is colored in shades of blue while the image destined for the right eye is colored in shades of red. Then the result will be: $R_{final} = R_{left}$; $G_{final} = 0$; $B_{final} = B_{right}$.

There are several types of anaglyph glasses in common use, red-blue, red-cyan, magenta-green and red-green. If the left and the right images are in grey then the RGB result could be: $R_{final} = Grey_{left}$; $G_{final} = (Grey_{left} + Grey_{right})/2$; $B_{final} = Grey_{right}$.

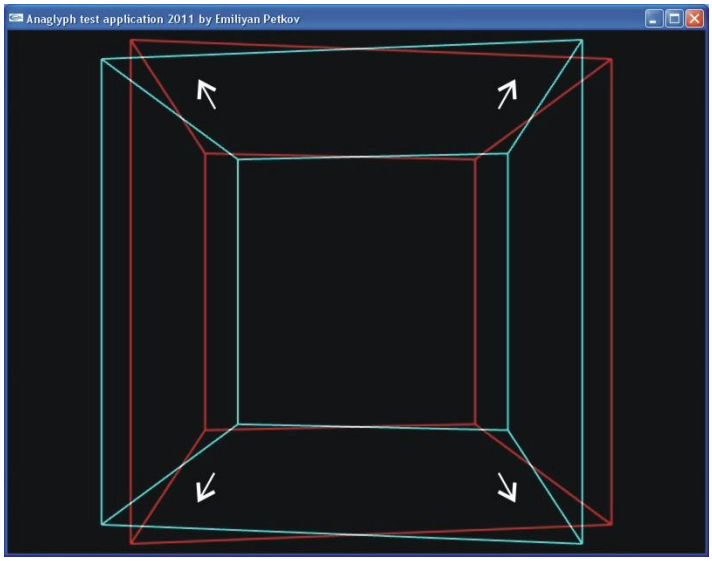


Figure 9. An anaglyph image generated by Anaglyph Test Application.
Convergence technique is presented

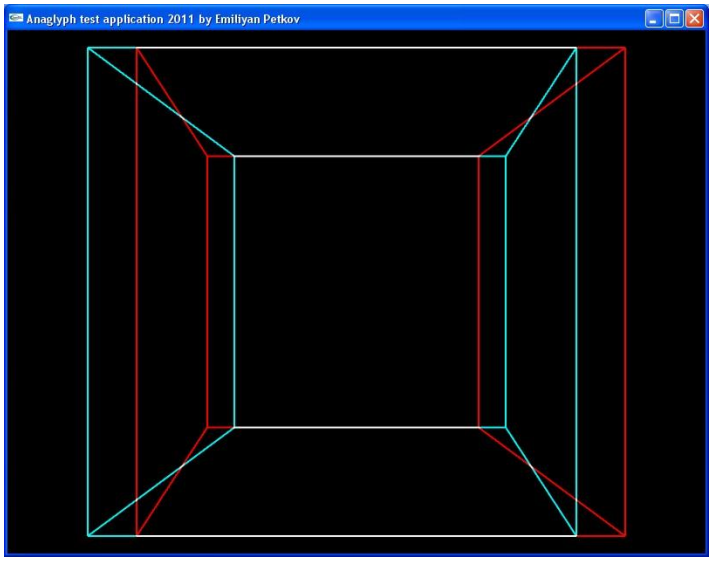


Figure 10. An anaglyph image generated by Anaglyph Test Application.
Parallel technique with asymmetric frustums is presented

If red-blue glasses are used the left camera contributes the red in the final image, the right camera contributes the blue in the final image, the green component of the final image is an average of the green channels from the left and right camera images: $R_{final} = R_{left}$; $G_{final} = (G_{left} + G_{right}) / 2$; $B_{final} = B_{right}$. Increasingly the standard glasses being used are red-cyan. The mapping commonly used is to take the red channel from the left camera image and the green-blue channel from the right camera image: $R_{final} = R_{left}$; $G_{final} = G_{right}$; $B_{final} = B_{right}$. The same can be accomplished but on a grey scale image: $R_{final} = Grey_{left}$; $G_{final} = Grey_{right}$; $B_{final} = Grey_{right}$.

To demonstrate and test the presented here research an anaglyph application has been developed. Its name is Anaglyph Test Application. It is OpenGL graphics application and a virtual 3D world has been created in it [3,10]. The two main techniques for creating stereo images are realized. It works in red-cyan color filters, so the images, that it generates, could be seen with red-cyan glasses.

In Figure 9 a realization of convergence technique could be seen. A stereo-camera system has been created with the following parameters of both cameras: w=800, h=600, fd=700, camera

separation cs=1/5.fd=140, vertical aperture γ=46.4º and horizontal aperture θ=59.5º. In the space there is a box and the convergence point of both cameras is in the geometric center of the box. In this image generated by the convergence technique vertical parallax is presented. It could be seen in Figure 9 (in every corner of the box). This results to distortions – in particular different height distortions that introduce artificial vertical disparities which can become uncomfortable to view.

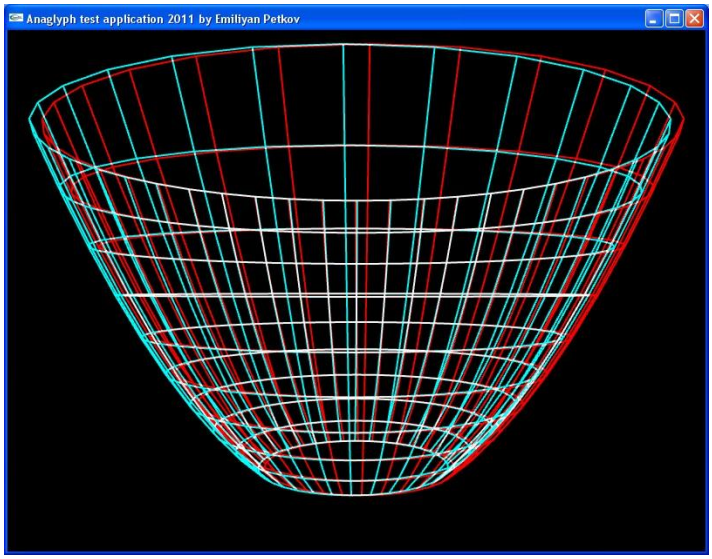


Figure 11. An anaglyph image generated by Anaglyph Test Application.
Parallel technique with asymmetric frustums over an Elliptic Paraboloid

In Figure 10 a realization of parallel technique with asymmetric frustums could be seen. A stereo-camera system has been created with the following parameters of both cameras: w=800, h=600, fd=700, camera separation cs=1/5.fd=140, vertical apertures γ1=γ2=23.2º and horizontal apertures θ1=25.24º and θ2=33.88º. In the space the same box as it is in Figure 9 is presented. In this image (Figure 10) generated by the parallel technique vertical parallax is not presented.

An anaglyph image generated by Anaglyph Test Application and parallel technique with asymmetric frustums over an object (Elliptic Paraboloid) placed in space between the stereo-camera system and the projected plane is presented in Figure 11. The images presented by Figure 9, Figure 10 and Figure 11 could be seen and depth effect be perceived only if they are color printed and the observer has red-cyan glasses.

All printed material, including text, illustrations, and charts, must be kept within the parameters of the 8 15/16-inch (53.75 picas) column length and 5 15/16-inch (36 picas) column width. Please do not write or print outside of the column parameters. Margins are 1 5/16 of an inch on the sides (8 picas), 7/8 of an inch on the top (5.5 picas), and 1 3/16 of an inch on the bottom (7 picas).

5. Conclusion

In this article techniques for stereo-images generation by 3D graphics applications have been discussed. One convergence technique and two parallel ones have been presented. Convergence method is may be more natural but has a big disadvantage, i.e. artificial vertical disparities that are introduced by vertical parallax. Parallel methods avoid the vertical parallax. Using symmetric frustums the two images should be trimmed off, so here we need trimming. Usage of asymmetric frustums avoids the need of trimming and it seems to be the best way for stereo images generation by 3D graphics applications. All calculations for setting up a stereo-camera system have been given as well. On the base of this research an anaglyph graphics application has been developed to

demonstrate and test the theory. Examples generated by this software to illustrate the calculations and the results of each technique have been given.

As future work the following thoughts could be shared: Usually human observes near objects more detailed as remote ones. And now some questions arise here concerning near objects: How to determine the limits of the virtual space? What limits should be placed when we allow 3D objects to pass out the window of the virtual world to the viewer? Can we achieve realism in terms of the visual objects (dimensions, distance, etc.)?

6. References

[1]. Agoston Max K. Computer Graphics and Geometric Modeling - Implementation and Algorithms. Springer, USA, 2005.

[2]. Baños Rosa M.Botella Cristina, Rubió Isabel, Quero Soledad, García-Palacios Azucena and Alcañiz Mariano. Presence and Emotions in Virtual Environments: The Influence of Stereoscopy. CyberPsychology & Behavior. New Rochelle, NY. February, 11(1): 1-8, 2008.

[3]. F. P. Brooks, „What's Real About Virtual Reality?," IEEE Computer Graphics and Applications, November/December, pp. 16-27, 1999.

[4]. Hill F. S. Jr., Stephen M. Kelley, "Computer Graphics Using OpenGL" – third edition. Pearson Education, Inc. Upper saddle river, NJ 07458, USA, 2007.

[5]. M. Roth, K. Tanaka, C. Weissman, W. Yerazunis. Computer Vision for Interactive Computer Graphics. IEEE Computer Graphics and Applications, May-June, pp. 42-55, 1998.

[6]. Mühlbach, Lothar; Böcker, Martin; Prussog, Angela. Telepresence in Videocommunications: A Study on Stereoscopy and Individual Eye Contact. Human Factors: The Journal of the Human Factors and Ergonomics Society, USA, Volume 37, Number 2, June , pp. 290-305(16), 1995.

[7]. P. May, „A Survey of 3D Display Technologies", The Society of Information Display (SID), USA, March/April, pp28-33, 2005.

[8]. Petkov G. Emiliyan. "One Approach for Creation of Images and Video for a Multiview Autostereoscopic 3D Display". International Conference on Computer Systems and Technologies CompSysTech'10, Sofia, 2010.

[9]. Salomon David. Transformations and Projections in Computer Graphics. Springer, USA, 2006.

[10]. Sherman William r., Alan B. Craig, "Understanding Virtual Reality – Interface, Application, and design. Morgan Kaufmann Publishers, an imprint of Elsevier Science, 340 Pine Str., San Francisco, CA 94104-3205, USA, 2003.

For contacts:
Assist. Prof. Dr. Emiliyan G. Petkov
Department of Computer Systems and Technologies
St. Cyril and St. Methodius University of Veliko Turnovo, Bulgaria
email: epetkov@abv.bg

MULTIWEBSITE SOFTWARE SYSTEMS

Dimitar Z. Dimitrov, Elena V. Racheva

Abstract: The paper describes new architecture and methodologies for mass website development. The proposed software system aims the establishment of development standards within software development companies which will optimize the development processes. This architecture will help upon building systems based on Software as a Service (SaaS) model. The development optimization the architecture will enable software development companies to serve and support times more companies then they can do with conventional development.
Keywords: Software Systems and Technologies, Software as a Service, Software Development Optimization.

Многосайтови софтуерни системи

Димитър Здр. Димитров, Елена В.Рачева

Резюме: В статията се описва нова архитектура и методи за създаване на масови Web-сайтове. Предлаганата софтуерна система цели внедряване в софтуерните компании на стандарти, които ще оптимизират процеса на разработването на сайтове. Тази архитектура ще помогне при изграждането на системи, базирани на модела SaaS (Software as a Service). Оптримизацията на архитектурата ще позволи на такива софтуерните компании да обслужват и поддържат в пъти повече фирми в сравнение с компании, използващи конвенционалния подход.
Ключови думи: Софтуерни системи и технологии, SaaS (Софтуер по поръчка), Оптимизация при създаване на софтуера.

1. Introduction

The definition “multiwebsite system” means a system which consists and controls a set of many websites. When analyzing software development technologies, new development frameworks impose standardization in development processes. Software design patterns as Model View Controller (MVC) are imposed as a main standard in web development frameworks.

This paper covers a description of architecture and methodologies for developing a software system for mass website development. The software architecture is a set of unique websites with similar functionality. Each website of the set can have a different owner and can be controlled separately. The system can have many website owners and each owner can have many websites. Each owner can control all of his websites from a single administration point using a single sign on (SSO). When the core functionality is created all the developers need to do is to develop the differences in each website. For example if all websites in the set has an average of 80% similarity in functionality, after building the core functionality based on the architecture to create a new unique website they build only the 20% difference. The other option is to have non-unique websites with unique content. In this case the core functionality will be prebuilt at 100%, and after the system is created all they need to do to create a new website is to add a new account for the new owner. That way, the multiwebsite software systems can optimize development processes to a level that small software companies can manage and support huge amount of clients in a certain field of development.

The paper describes a method to convert any regular single website MVC architecture to a multiwebsite software system. This type of system architecture can implement specifics of the

Service Oriented Architecture (SOA) [1]. The multiwebsite system architecture is also inspired by other well known web architectures [2].

2. Architecture for a Multiwebsite Software System

There are two ways of creating a multiwebsite system. It can be created from scratch or a single website can be converted to a multiwebsite system. The right approach should be chosen depending on the specifics of each project. If a company decides to build Software as a Service (SaaS)[3] solution based on the multiwebsite architecture it can build it from scratch to best meet the architecture requirements. If the company has already created a single website, which will represent a node of the multiwebsite set, they should choose the conversion from a single website method to a multiwebsite system.

The case study example will be based on MVC architecture as it has the Model abstraction to the database which is a good example for the database queries conversion methods. Web sites created using an inline queries are harder to convert, but can be done using the same guidelines.

3. Conversion Method from a Single Website to a Multiwebsite Software System

The method can be described as the following series of steps:

a. The MVC architecture of the original single website

When a website is built using MVC based framework the architecture is as described on Figure 1 [4]:

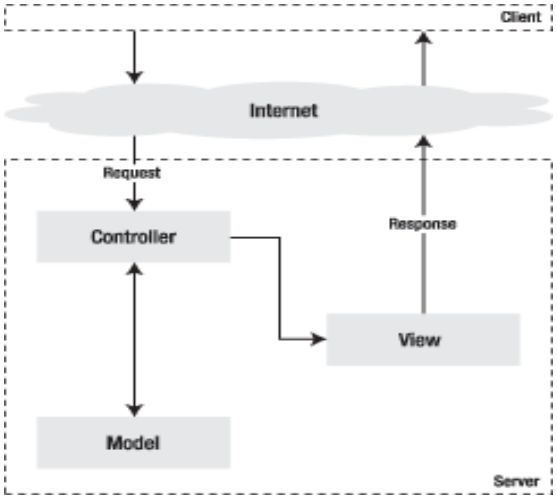


Fig. 1. Model View Controller Architecture

Converting MVC based website can show clearly how each part of the software should be manipulated. For creating the simplest multiwebsite system, the Model of the single website is the first layer which has to be converted. The Model is an abstraction of the database. The conversion should result in the ability of all websites of the system to have their own database records in the tables the original website had. This will allow all websites to have unique content.

b. The website nomenclature table

The core of this method is the website nomenclature table in the database. A table with the website main settings needs to be created. The system websites only represent different versions, views and content with the same functionality depending on the user's entry point.

The website nomenclature in its simplest form should contain an identifier column and a website name column for visual recognition within the administration. The identifier column will be related to each table in the database. What the user gets as a response will be depending on the

website identifier he submits. The submission of the identifier is transparent for the end user. The easiest way is to have the id directly posted with the system url request e.g. http://system-domain.com/8. With the right routing rules the identifier will be set for the desired website.

c. The multiwebsite system initialization

To get things really transparent a new "domain" column has to be added to the website nomenclature table. That way any website can be defined by a certain domain name. For example when a user requests the url http://mydomain8.com the multiwebsite system initialization will handle the request and will find the identifier which this domain stands for. Once the system gets the identifier for a website it will bring all the content, views, settings and custom functionality for that identifier. In order all applications to have access to the chosen identifier the initialization of the website has to put it in a session variable to be available through the whole user session. The initialization should be executed whenever a session expires or a new session has to be created for a new user. For further use in this paper the site identifier variable can be named $site_id.

If the system will use unique domain names for each website, each new domain should be forwarded to the multiwebsite system front controller where all the initialization and dispatching work will be done. This means that each domain will open the same physical location but the user will get different resources based on the domain name. For example there can be http://shopsite.com, http://modernshops.com, http://shopformen.com all opening the same system, but the user will get transparently 3 unique websites.

The initialization of the administration of the website should be done separately. The principals of identification of the website in the administration are similar to those in the front end. The difference is that an administrator should be able to control more than one website. Depending on the administrator rights, when he logs in the administration panel the initialization of the system will find the administrator depending on the credential details like username and password. Then depending on his rights over certain websites the system will allow him to reinitialize for different websites. The initialization will show him the administration for his default website and will give him an option that could be a drop down menu with the websites he has access to. When he chooses a different website from the menu a re-initialization will occur with the new website identifier.

d. Conversion of the Model layer

Once the website identifier is defined, the multiwebsite system must distinguish the right resources for the identified website. The content the user will receive will be delivered trough the Model layer. All the queries to the database are in this layer. All methods in the Model need to be modified to query only the data of the identified website. To reach this goal a new site_id column must be added to all tables in the database that will be shared between the websites of the system. This column will be related to the website identifier column in the website nomenclature table. The example in Figure 2 shows a relation of a product table from an ecommerce website to the website nomenclature table.

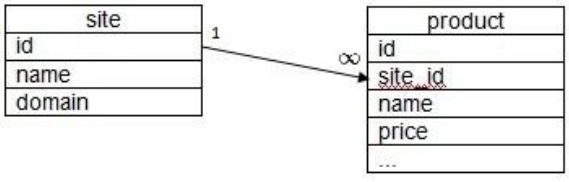


Fig 2. Example of a structure and relation of the website nomenclature table to a random table in the database

When the physical database is modified and the model is rebuild according to the changes, the next step is to modify the methods in the Model that query the database. The "SELECT" queries must have additional clause to the "WHERE" statement with the website identifier. E.g. "SELECT

* FROM products WHERE site_id=$site_id". The "INSERT" and "UPDATE" queries must be modified with an additional "SET" statement. E.g. "INSERT INTO products SET price=$price, name=$name, site_id=$site_id".

Some development frameworks use separate sub-frameworks for the Model layer. These sub-frameworks implement the Object Relational Mapping (ORM) technique. More popular ORM projects for PHP are for example Doctrine and Propel. If an ORM system is used in the original website the conversion methods should be done according to its rules and object interconnectivity. The result queries should comply with the conversion rules described for standard SQL queries.

e. Administrators (owners) of websites in the multiwebsite system

The conversion in the back end of the system is the same as the front end. The difference is that there is an authentication with administrator's credentials. Based on this credentials administrators will have the right to view and work with the administrator's section or just part of it, depending on the rights he has. He can have read only rights, or just permissions for a number of modules and actions. When an administrator of a website has access permissions only to a single website, he will not see any difference then administrating the original single website. But if he has access to more than one website he will enjoy the benefits of the Single Sign On (SSO), the common administration user interface of all websites he manages and the possibilities of interconnectivity between his websites resources.

In a multiwebsite system there is a need to manage and support the whole system. This means a super administrator access needs to be created. When we have implemented the current logic this is quite simple. We just create a user with access to all websites. This will grant them permissions to help site owners with their daily work and troubleshoot problems. Depending on the type of confidentiality of specific systems this type of access can be limited.

The super administrators need a core functionality, to work with website accounts and administrator permissions. This functionality will allow them to add/approve new websites to the multisite system, stop/start websites depending on payment status, terms and conditions compliance, or other rules.

If the owners can add their own websites to their account it will increase the system's automation. If the websites in the system are paid, a payment gateway can easily automate that and make the whole website creation process unsupervised. This will free the man power needed to create each account. And those people can focus on support and upgrades development.

At this point the multiwebsite system will be completely functional. Each website can have unique content, one owner can have many sites and one site can have many owners/administrators. All parts of a website can be multiplied in a multiwebsite system and all websites in the multiwebsite system can be unique to the users.

f. Conversion of Views and Templates

Again the separation of the View in the MVC architecture is a good work practice and will enable us to use different approaches for the conversion of the different layers of a website. In order to make unique designs to cover the similar functionality, the templates, views and all publicly available files like images and css scripts must be separated in a group. This group can be a folder with a specific name, which can be cloned and changed in order to create a brand new unique design for a website in the multiwebsite system.

In order to make things more dynamic new rows "template name" and "template path" should be added in the website nomenclature table. This way the different available templates can be controlled trough the system administration. If the grouped templates are in different locations and with different names as suggested, changes in all the controllers have to be made, and the controller templates should be decided depending on the website configuration.

If the software system is developed using a Templating Framework, the previous operations will be unneeded as those systems include much more sophisticated ways to allow designers to

develop templates for a system and ways for users to manage them. There are many projects for template management like Twig, TinyButStrong, Outline, Open Power Template, etc.

The template is one for the whole user session, therefore it is a good practice to define each website's template in the initialization part and store the defined result in a session. This way all controllers will set the right template name and path stored in the session and no database calls will be necessary after the initialization of a website.

A multiwebsite system can be developed with prebuilt designs which can be used by different website owners. The designs can have free and paid versions. Website owners should be able to create and upload their own designs based on any of the original designs. This way each site owner can have a unique website just by tweaking an original design.

For the more professional designing a interface protocol should be distributed to the website owner with guidelines of the requirements needed in order to create a brand new custom design.

g. Converting the Static Content

The only part of a single website that is left for conversion is the static content, hard coded in the web pages. If we don't convert this hard coded text it will not fit all web sites in the system. Suggested approaches for making static content dynamic are the following:

- Importing all static content in a database. Creating a new model with all the model requirements commented in point 4. The model needs to have a row identifier, which represents the placeholder of the static content. Then all the static content should be replaced with the corresponding in the database according to the content identifier and site identifier.
- Another option is using a unique template. If the static content is hardcoded within a template, it can be easily customized if the template is cloned for customization. This approach is not as flexible as the new template will need additional support resources.

h. Dividing the system into modules for upkeep and management optimization

Enabling site owners to develop their own unique functionality could be achieved by separating the whole system into logical modules. A new Module nomenclature should be created. The modules can have hierarchy of parent-children modules. They can be called by module identifiers in the system websites. The system administrator can start or stop a module for each website. If the system is module based it will not be strictly typed and all type of modules can be added. E.g. there can be a module for e-commerce and a module for e-learning. And two different types of websites can be joined in one multiwebsite system. If the multiwebsite system developer decides to allow the website owners to build their own modules, a module API for the module development should be created. Dividing a system into modules is a well known strategy to make an easily extendable and upgradable software [5].

i. Scalability of a multiwebsite system

The multiwebsite system can start with one website and in time reach a million websites with a rapid grow. With each additional website more resources will be needed. The rules for scalability [6] and optimization of a multiwebsite system are not different then a regular rapidly growing website. Techniques like load balancing, Database clustering and partitioning, etc. are just as applicable in a multiwebsite system as any other fast growing software system. A new trend in recent years is focused around Cloud Computing [7].

4. Conclusions and Future Work

The described conversion method has been successfully implemented in an experimental project. A multiwebsite ecommerce platform has been created with the trademark Get a Mall http://getamall.com. The project offers online shops as a facebook application. The application is

installed as a tab menu in a desired facebook page. The project has 4 initial unique designs, and some additional paid modules as group shopping.

The theoretical and practical experiments in the development of a multiwebsite system architecture has led to the following conclusions:

- The multiwebsite system architecture can decrease the resources needed to handle a big amount of orders, and to support a big amount of customers.

- It can bring the focus of software companies more on the business side to optimize sales and marketing strategies, rather than focusing on the development side to recreate similar websites over and over again using valuable man power and other resources.

- As most SaaS systems it will bring much less risk in day-to-day business. Because of the monthly or annual fees it allows to make better strategic planning for the companies as they can predict incomes and expenditures better.

- It will make the company developing the system much more competitive. E.g. a customer wants a unique ecommerce website. With the system it can be done within a minute with prebuilt designs, or just by tweaking a design, while standard unique development can take weeks or months.

- By reducing the cost for development the company can also be competitive in pricing.

- To create the original ecommerce application a team of 2 developers and one designer spent all together 480 hours. To implement the conversion method the same team spent an additional 780 hours. The system now allows a website with the same functionality to be created within a minute of a non-developer using the administration panel. More precise data could be collected after more executed projects, but with the initial data we can conclude that this method should be used if the company intends to build more than 3 similar websites. The reason for this conclusion is that the time to create the system took nearly the time to create 3 websites of this scale. The data is relative and results will not be the same for the different type of projects.

In the near future a new experiment will be conducted upon new theory optimizations of the method. A standard prebuilt administration will be developed to be used freely for all projects implementing this method. This will help save the developers a lot of initial work, which is mandatory for this type of architecture.

Acknowledgments: The carried out research is realized in the frames of the project BG051PO001-3.3.06-0005, Program **'**Human Resources Development'.

References

[1] M. Rosen. Applied SOA: Service-Oriented Architecture and Design Strategies – Wiley, 2008
[2] L. Shklar. Web Application Architecture – Wiley, 2009
[3] K. Roebuck. Saas - Software as a Service: High-impact Emerging Technology - Tebbo, 2011
[4] P. Vora. Web Application Design Patterns - Morgan Kaufmann, 2009
[5] K. Roebuck. Engineering Long-Lasting Software - Strawberry Canyon LLC, 2012
[6] M. L. Abbott. The Art of Scalability - Addison-Wesley, 2009
[7] B. Sosinsky. Cloud Computing Bible – Wiley, 2011

For contacts:
M.Eng. Dimitar Z. Dimitrov, PHD student
Department of Computer Science and Technology
Technical University – Varna
E-mail: dimitrov@varnasoft.com
Accos. Prof. Elena V. Racheva, PHD
Department of Computer Science and Technology
Technical University – Varna
E-mail: elracheva@yahoo.com

СИСТЕМА ЗА АНАЛИЗ И ДИАГНОСТИКА НА ЦИФРОВИ ИЗОБРАЖЕНИЯ НА КРЪВНИ ПРОБИ

Венцислав Г. Николов, Христо Г. Вълчанов

Резюме: В статията е представен подход за изграждане на система за бърза диагностика на заболявания, засягащи състоянието на кръвните клетки като промени в големината, формата, оцветяването, наличие на включвания в тях и др. Представеното решение се базира на анализ на изображенията на кръвните клетки чрез отделяне на техните контури и тяхното използване за формиране на описание и обучение на невронна мрежа. Системата позволява систематизиране на данните за разпространението на заболявания, което дава възможност за изграждане на базирана на интернет технологиите система за ранно известяване при епидемии.
Ключови думи: Невронни мрежи, диагностика на кръвни проби, обработка изображения.

System for analysis and diagnosis of digital images of blood samples

Ventsislav G. Nikolov, Hristo G. Valchanov

Abstract: In this paper is described an approach for system for facilitation of disease diagnosis that concerns the red blood cells state as changes in their form, color, foreign substances, etc. The present solution is based on analysis of blood images by separating of blood's edges and their use for creating a description and training of a neural network. Moreover it allows systemizing data for disease dissemination that facilitates development of a global Internet-based early epidemics warning system.
Keywords: neural networks, blood images diagnosis.

1. Увод

Автоматизираният визуален анализ на изображения намира все по-широко приложение в софтуерните системи. Той има предимствата, че осигурява висока скорост и значително намаляване на броя на хората, ангажирани по извършването на дейности, свързани с анализа.

Редица заболявания които водят до морфологични изменения на кръвните клетки могат да се идентифицират с помощта на визуален контрол [8]. Такива са например, някои видове малария, анемия и др., при които се наблюдават промени във формата, размера или други характеристики на кръвните клетки, настъпили поради външни включвания или патологични изменения. Системите за автоматичен анализ на кръвни проби преодоляват бавната скорост на човешкия анализ, отнемащ до няколко часа. Освен това, в случай на епидемии, се налага бързо диагностициране на широк спектър пациенти, което изисква наличието на голям брой медицински лица. Този брой в икономически изостаналите географски региони обаче, е често недостатъчен. Разработването и внедряването на автоматизирани системи, които позволяват в кратки срокове с минимални човешки ресурси да подпомагат диагностиката, могат да позволят увеличаване на скоростта на анализа и да намалят влиянието на човешкия фактор.

В статията е представен подход за изграждане на система за бърза идентификация на заболявания, при които се наблюдават морфологични промени в червените кръвни клетки. Идентификацията се извършва посредством анализ на изображения получени по стандартен USB интерфейс от свързана към микроскоп камера за наблюдение на кръвните проби. Системата се характеризира с малки изисквания по отношение на техническото оборудване -

могат да се използват обикновена уеб камера и неспециализиран микроскоп в състояние да увеличава около 1000 пъти картината от кръвните проби.

Известни са редица софтуерни решения използващи различни методи за анализ на цифрови изображения от кръвни проби [10], като например трансформации на Хаф за окръжности [12], хистограми и средни отклонения на сегменти в изображението, параметрични описания [11] и др. Представеното в статията решение се базира на анализ на изображението чрез отделяне на контурите на кръвните клетки и тяхното използване за формиране на описание и обучение на невронна мрежа. Обучението на мрежата включва и последващо класифициране на нови примери на променени кръвни клетки.

2. Описание на подхода

2.1. Обработка на изображенията

В общия случай микроскопските изображения на кръвните клетки съдържат голям обем информация, което налага използване на техники за бърза обработка и анализ. Целта е да се извлекат т.нар. непроизводни елементи [1], които да предоставят достатъчно пълно описание. Тъй като изображенията на кръвните проби са с голям брой визуални обекти, които трябва автоматично да се анализират, се налага извличането само на определени характеристики от сегментите в изображението. За целта се съставят признаци, всеки от които представлява апроксимационна права линия на контура на кръвна клетка. Един признак се представя чрез вектор от пет елемента (x_1, y_1, x_2, y_2, sin, cos), като за тяхното определяне се изпълняват следните стъпки:

- цифрово представяне на изображението;
- определяне на контурите на визуалните обекти (червени кръвни клетки) и тяхното изтъняване;
- обхождане на контурите и определяне на контролни точки за обособяване на признаци;
- формиране на описание на обектите в изображението чрез описание на признаците.

2.2. Цифрово представяне

От полученото изображение първоначално се формира двумерна матрица от елементи, всеки от които се състои от четири стойности, съответстващи на силата на трите цветови компоненти (червена, зелена и синя) и степента на прозрачност на съответния пиксел. За да се извършат операциите в следващата фаза, необходимо е всеки елемент на изходната матрица да се третира като атомарен елемент. Това налага трансформация на цифровото представяне по такъв начин, че всеки пиксел да се характеризира само с едно число, съответстващо на неговата яркост. За целта се извършва преобразуване на цветното изображение в черно-бяло. Алтернативен подход е да се обработва изображението във всяка от цветовите компоненти и комбинирането на резултатите след обработката. Формирането на черно-бяло изображение в системата се извършва чрез пресмятане на средната стойност на яркостта на всяка цветова компонента.

2.3. Определяне на контурите

Определянето на контурите се базира на откриването на големи градиентни промени в яркостта на черно-бялото изображение [4]. За да се открият такива промени се пресмята производната на яркостта. При едномерно пространство, това се извършва по следния начин:

$$f'(x) = \lim_{\Delta x -> 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \quad (1)$$

При разглеждане на двумерното пространство, в представената система, се съставя оператор на Превит [2] за откриване на контури в изображението по хоризонтала, вертикала и по диагонал. Изображението се обхожда с конволюционна маска, при което за всеки пиксел с координати (*i, j*) се извършват изчисления, аналогично на [9]:

$$g(i, j) = \sum_{m=-a}^{a} \sum_{n=-a}^{a} f(m, n) h(i - m, j - n), \quad (2)$$

където $a=(k-1)/2$, f е яркостта на изображението, а h е конволюционна маска с размери k x k.

Формула (2) описва формирането на коефициентите, оценяващи наличието на контур.

За различните посоки се съставят различни маски и когато силата на градиентния скок, определен чрез формула (3), е по-голяма от предварително зададен праг, в съответната точка се маркира наличие на контур, което се използва от следващите етапи на обработка.

$$magn(\nabla f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} = \sqrt{M_x^2 + M_y^2}, \quad (3)$$

където

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x} \quad (4)$$

$$\frac{\partial f}{\partial y} = \lim_{\Delta y \to 0} \frac{f(x, y + \Delta y) - f(x, y)}{\Delta y} \quad (5)$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \quad (6)$$

Посоката на контура се определя по следния начин [5]:

$$dir(\nabla f) = \tan^{-1}\left(\frac{M_y}{M_x}\right) \quad (7)$$

2.4. Обхождане и формиране на описание

Описанието на кръвните клетки в картината на изображението се формира, чрез следните стъпки:

- обхождане на контурите [6]. Пикселите, принадлежащи на контур, се разглеждат като върхове на граф и се изпълнява обхождане в дълбочина с отброяване на посетените върхове в брояч.

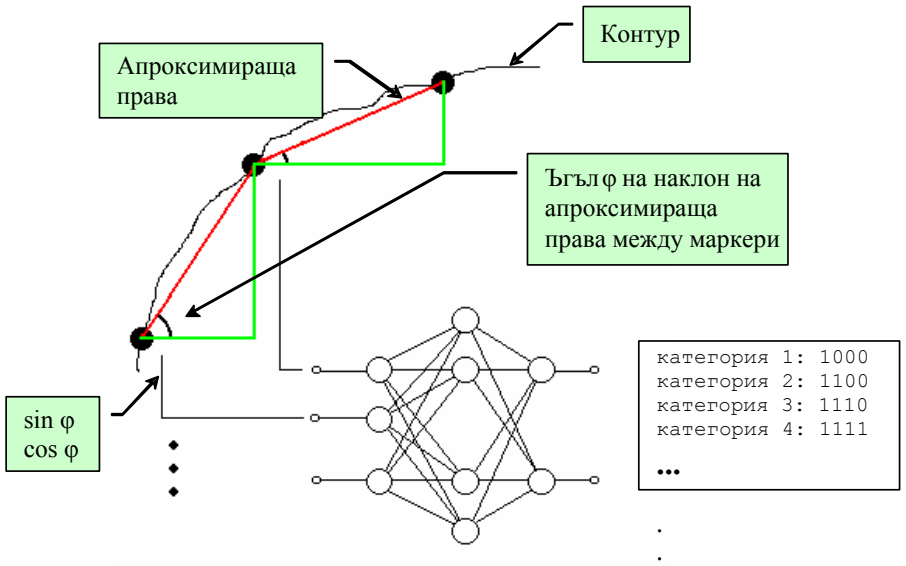
```
checkLineLength((x, y), дължина){
   пиксел_(x, y)_се_маркира_като_обходен;
   ако_съсед_на_(x,y)_е_контур_и_не_е_обходен
```

```
        {
          дължина = checkLineLength(съсед, дължина+1)
        }
      return дължина;
    }
```

Ако стойността на брояча е равна на предварително зададена максимална стойност или е достигнат край на линията на контура, се поставя маркер на контролна точка.

- между съседните контролни точки се построяват апроксимиращи прави (Фиг.1). Всяка права *i* се определя еднозначно, чрез координатите на двете крайни точки – (x_1, y_1, x_2, y_2). Освен тези координати, за всяка права се определя и ъгъла на наклона φ спрямо абсцисата;
- пресмятат се синуса и косинуса на ъгъл φ;
- окончателно се формира описанието на признак *i* чрез елементите (x_1, y_1, x_2, y_2, $sin(\varphi)$, $cos(\varphi)$).

От информацията, описваща признаците, се използват само последните два елемента - $sin(\varphi)$ *и* $cos(\varphi)$.

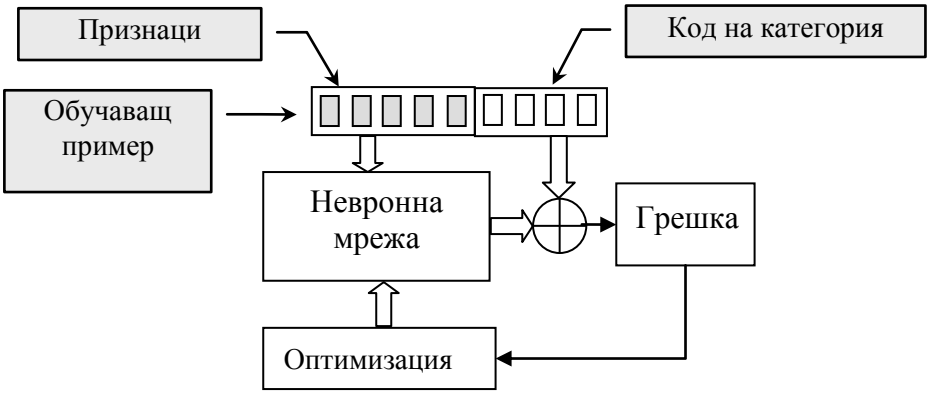


Фиг. 1. Извличане на информация от признаци в изображението за обучение на невронната мрежа

Това позволява да се пренебрегне размера на увеличението на кръвните клетки от камерата и да се извърши анализ на формата на клетките или наличието на чужди тела в тях. Тези данни освен това приемат стойности само в интервала -1 ÷ 1, което е основно изискване към данните за обучение на невронни мрежи с обратно разпространение на грешката [3]. Използват се и синус и косинус, тъй като ако някоя от апроксимиращите прави е хоризонтална или вертикална права, то синусът (респективно косинусът) е нула, което не носи полезна информация за анализа.

3. Обучение на невронната мрежа

След формиране на описанието на контурите на изображенията се пристъпва към обучение на невронната мрежа – фиг.2. Тя е от тип многослоен персептрон и се обучава по широко използвания алгоритъм с обратно разпространение на грешката.



Фиг. 2. Обучение на невронната мрежа

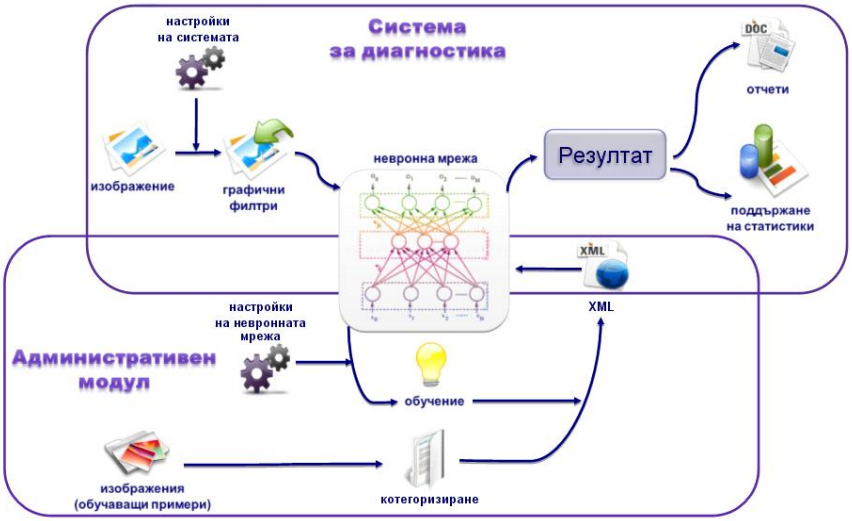
Броят входове е равен на броя на всички признаци умножен по две (за синус и косинус на всеки признак), а броят на изходите е равен на броя на категориите. Системата поддържа множество възможности за настройка: различни по тип активационни функции, критерии за край на обучението и т.н. След успешно обучение, при достигане на достатъчно малка грешка и валидиране на способността за генерализация, целият контекст (тегла и настройки на невронната мрежа) се съхраняват в XML дърво. При заявка от клиент, дървото се предава към него, където модулът, след построяване на невронната мрежа в паметта, може да я използва за класификация на нови изображения.

Входно-изходните обучаващи двойки вектори се формират по следния начин:

- входният вектор на всеки обучаващ пример се определя от признаците на съответното изображение. Те се формират след определяне на контурите на кръвните клетки;
- изходният вектор на обучаващия пример се определя, чрез кодиране на съответната категория към която принадлежи изображението [7]. Кодовете на категории са двоични вектори, във всеки елемент на които се записва стойност единица, ако индексът му е по-малък или равен на поредния номер на текущата категория или нула в противен случай.

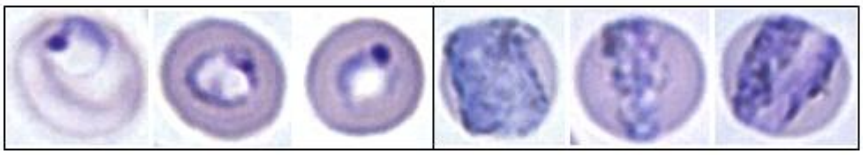
4. Реализация на системата за диагностика

На фиг. 3 е показана структурата на разработената на базата на предложения подход система за диагностика. Тя се състои от административен и множество клиентски модули.



Фиг. 3. Структура на системата за диагностика

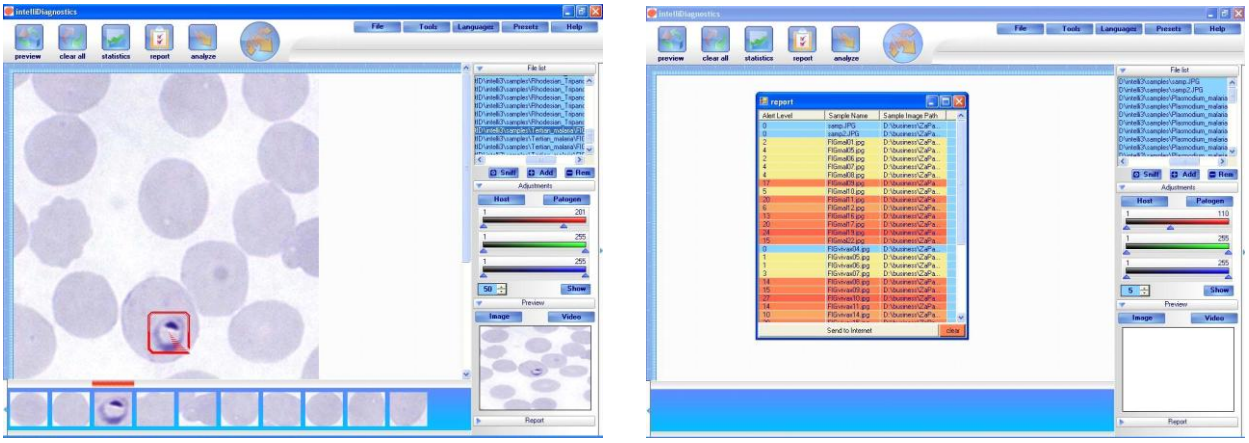
В административния модул се съдържа информация за заболявания, която се предоставя към клиентските модули с помощта на услуги. Информацията се изгражда чрез предварително структурирани и групирани според заболяванията изображения на кръвни клетки. Класифицирането се извършва от експерти-медицински лица, които предоставят примери за използване (Фиг. 4).



Фиг. 4. Кръвни клетки, класифицирани в две примерни категории на заболявания

След получаване на изображенията, информацията за тях се представя чрез конекционисткия подход с използване на еднопосочна невронна мрежа, чиято топология и параметри на обучение се определят според представените примери.

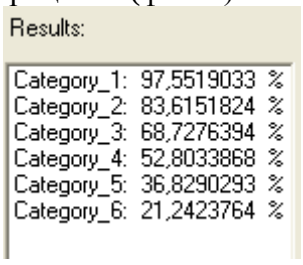
Клиентските модули (фиг. 5) могат да инициират заявка за последните дефиниции на заболявания и да извършат класифициране на подадено изображение. В резултат от заявката, главният модул предоставя обучена невронна мрежа във вид на XML дърво, която се построява в паметта при клиента. XML информацията съдържа архитектурата на невронната мрежа (брой слоеве, брой неврони) и самите тегла на връзките. Функциите на клиентския модул позволяват единствено диагностика посредством определяне на категорията на представеното на входа изображение.



(а) (б)

Фиг. 5. Анализ на пробите – индивидуална проба (а), пакетен режим (б).

Тъй като подаваните на клиентския модул изображения на проби могат да са с различно качество, преди прилагане на алгоритмите за анализ, се прилагат филтри за цветове, настройки за яркост и др. Към изображението се извършва сегментация, обектите се отделят и се анализират с помощта на заредените от административния модул дефиниции. При откриване на патологични изменения се извършва диагностика на съответните клетки, като наличието на заболяване се посочва в проценти (фиг. 6).



Фиг. 6. Резултати при откриване на заболяване

5. Заключение

Представената система използва конекционистичен подход за класификация на изображения, представени във вид на матрици от числа, с използване на информацията за предварително формирани класове от изображения. Предстоят експериментални изследвания на чиято база да се определят оптималните параметри при обработката на изображения и обучението на невронната мрежа. Точността на класифицирането се определя най-вече от броя на представените примери и тяхното правилно разпределение по категории на заболявания. Изграждането на знанията за проблема на едно централно място и мултиплицирането на клиентските модули за използване на тези знания съкращава времето за анализ и дава възможност за висока степен на автоматизация. Така по естествен начин експертните знания на медицинските лица могат автоматично да се разпространяват и използват на много места, въпреки географската им отдалеченост.

Цел на бъдеща работа е интегрирането на представената система в цялостна инфраструктура за ранно известяване при епидемии. Тази инфраструктура се предвижда да бъде изградена посредством мобилни устройства, върху които функционира системата за диагностика и облакови услуги, предоставящи отдалечен достъп до централна база с натрупани диагностични знания за заболявания.

Литература

[1]. Гочев, Г. Компютърно зрение и невронни мрежи, С., изд. на ТУ–София, 1998, стр.251.

[2]. Павлова, П. Цифрова обработка на изображения. П., изд. на Фондация "Физика, инженерство, медицина – XXI", 2005, стр. 64.

[3]. Fausett, L. Fundamentals of neural networks. Architectures, algorithms and applications. Prentice Hall, 1994, pp. 461.

[4]. Acharya, T., A. K. Ray. Image Processing: Principles and Applications, Wiley interscience - Ney Jersey, 2005, pp. 425.

[5]. Trucco, J. CS491E/791E: Computer vision Lectures. Department of Computer Science, University of Nevada, Reno, NV 89557, 2004.

[6]. Jahne, B. Digital image processing, Springer-Verlag, 2002, pp. 598.

[7]. Theodoridis, S., K. Koutroumbas. Pattern recognition, Elsevier, 2008, pp. 984.

[8]. Meyer-Base, A. Pattern Recognition for Medical Imaging, Elsevier, 2004, pp. 386. Dougherty, G. Digital Image Processing for Medical Applications, Cambridge University Press, 2009, pp. 462.

[9]. Jambhekar, N. Red Blood Cells Classification using Image Processing. //Science Research Reporter, 2011, Issue 3, pp. 151-154.

[10]. Bronkorsta, P et al. On-line detection of red blood cell shape using deformable templates. //Pattern Recognition Letters, 2000, vol.21(5), pp. 414-424.

[11]. Victorian Bioinformatic Consortium. http://www.vicbioinformatics.com (последно достъпна 12.2011).

За контакти:
гл.ас.д-р инж. Христо Г. Вълчанов
катедра „Компютърни науки и технологии"
Технически университет - Варна
E-mail: hristo@tu-varna.bg

д-р инж. Венцислав Г. Николов
E-mail: v.g.nikolov@gmail.com

INTERACTIVE MULTIMEDIA TOOLS FOR ONLINE EDUCATION IN POWER ELECTRONICS

Angel St. Marinov

Abstract: The current paper suggests a set of interactive multimedia tools for presentation of complex educational material in power electronics. The tool set is developed in flash based application and was implemented for both lecture presentations and e-learning. The necessity of such tools is explained by defining basic problems and issues related to presenting materials in power electronics. The means to resolving these issues which can be accomplished through the use of interactive software, flash based animations are presented. An example with a dedicated application for power electronics is given.
Keywords: power electronics, electronic education, interactive tools.

ИНТЕРАКТИВНИ МУЛТИМЕДИЙНИ ПРИЛОЖЕНИЯ ЗА ЕЛЕКТРОННО ОБУЧЕНИЕ ПО ПРЕОБРАЗОВАТЕЛНА ТЕХНИКА

Ангел Ст. Маринов

Резюме: В настоящата статия са представени набор от интерактивни интернет базирани приложения. Тези приложения могат да бъдат използвани като презентации в част от лекционен курс или като инструмент за дистанционно обучение. Дискутирани са основните проблеми при представянето на материали за електронно обучение. Представени са основните средства за реализирането на тези приложения като интерактивен софтуер и флаш базирани анимации. Представен е пример на готово приложение.
Ключови думи: силова електроника, електронно обучение, интерактивни приложения.

1. Introduction

The term Power Electronics refers to a specific branch of the electronic technologies. This branch involves the conversion of electrical energy through the utilization of solid state semiconductor switches [1]. Power electronics are an important feature of every modern device powered by electrical energy. Some major application that can be mentioned include: (i) power supplies - basically for all modern electronic devices; (ii) control of energy generation - mostly in smart energy systems and renewable energy generation; (iii) energy conversion for industrial processes - induction heating, welding, electrical motor control, etc..[2].That defines power electronics into a required subject within higher education programs and curricula – under one form or another - for each engineer that specializes in the professional field of Power engineering, Electronics and Automation (professional field given defined by Bulgarian legislation [3]).

Currently with the improvement of computer technologies – online and remote studies become a popular option for higher education in many fields, including engineering. That poses various challenges in presenting and adapting knowledge and information for remote users. This is especially true for the subject of power electronics where complex information and data is presented through various block and circuit diagrams, waveforms, equations and text.

The current paper: (i) addresses some of those challenges by presenting issues and problems that were noted through analysis and derived from practical experience in online education in power electronics (Section 2. Presentation and adaptation issues for online education in the subject of power electronics); (ii) gives motivated suggestion for solutions of the noted issues (Section 3. Presented solutions for materials for online education in the subject of power electronics); (iii) gives

an example of the presented solutions (Section 4. Example application); gives a summary of the presented work (Section 5. Conclusion)

2. Presentation and adaptation issues for online education in the subject of power electronics

By analyzing the content of the initial source material for power electronics and its specifics, several basic types of issues can be distinguished. Those issues were defined based on a dedicated literature review on online education [4, 5, 6] and the experience gained through the conduction of two projects within the Technical University of Varna [7, 8, 9]. The projects include elements of online education and have courses relevant to power electronics. The issues can be summarized as follows into a three main groups:

a. *Difficulties in presenting of information in a comprehensive way that explains the correct processes in the circuit.*
   The purpose of each element in a power electronic circuit is to direct the electrical flows in order to convert energy for the use of specific electrical device. The main difficulty in explaining the operation and mechanics behind power electronic circuits lies in properly presenting the change of current or voltage and relation between the used component, defined formulas and obtained waveforms. This is especially problematic in online education, where the user has to discover the relation between circuitries, waveforms and equations by himself.
b. *Difficulties of providing simultaneous display of the equivalent circuits formed by the commutation of the electronic switches.*
   Other significant problem in e-learning materials for power electronics is to properly and clearly define the path of the current for a specific state of the components in a circuit and the change of path when related to a specific state of some components. In conventional teaching materials those relations are presented trough equivalent circuits, where each state of the main circuit is presented by a separate equivalent circuit. In online education this however requires multiple figures and further explanation, which would require multiple presentation screens that can confuse the user.
c. *Difficulties of representing complete information while maintaining a scroll free screen.*
   When presenting information for online education it is important to hold the attention span of the user. When various information including figures, diagrams, circuits and text is presented, multiple computer screens are required. This can be obtained either by scrolling or by switching between screens. If the material cannot be divided into small sections, this could distract the user, since for the same section text and figures have to be scrolled up and down or backtracked trough different screens. This is especially true for power electronics where the explanation of a single circuit could hold numerous figures, equations and large paragraphs of text.

3. Presented solutions for materials for online education in the subject of power electronics

Based on the issues mentioned above several approaches that can resolve them were developed and are suggested. The approaches include the utilization of modern computer based multimedia that can integrate animated features and user interactivity. In the case with examples that follow in the next section the computer multimedia is based on flash, HTML5 or other popular technologies can be used as well. These approaches can be summarized as follows into three defined solutions:

a. *Synchronization between waveforms, equivalent circuits transitions and explanations*
   The presented solution includes a simultaneous presentation of equivalent circuits, waveforms and process description. In order for the solution to work all of the components

of the information have to be synchronously presented through animation. An especially important part is the synchronism between the waveform animation and the change rate of the equivalent circuit. This allow the user to see the relation between the equivalent circuits (defined by the state of the electronic switches) and the change of the voltages and currents within the topology. The combination of all the components and their simultaneous presentation also allows the user to concentrate on the given application – information is presented in a single screen, scrolling and switching between screens can be avoided.

b. Presentation through multiple tabs on a single screen

The multiple components in a power electronics schematic could be presented on several subplot graphics with color code in order to easily present the path of the current, the change of voltage and certain switching state. This again leads to concentration of the information and a more comprehensive presentation that will not require the user to scroll or backtrack through different screens.

c. Comprehensive control of parameters that affect the circuit functionality

In every interactive presentation there should be a number of functional buttons, which control several important circuit parameters. The control of the parameters allows viewing different modes of operation within a single application. It will also allow the user to interact with the circuit adding an exploration element.

4. Example application

The presented in the previous section solutions are given as a summary in an example application. The example application involves a short course for controlled rectifiers. It combines three different types of rectifiers – single phase controlled bridge rectifier, single phase half wave controller rectifier and single phase controlled rectifier with centered – tapped transformer. The different types can be selected through invective buttons - fig.1. By clicking on the chosen schematic another slide opens as presented in fig.2 containing the related schematic (1) and two other areas, one for the parameters, equations and descriptions (2) and the other for the waveforms in defined test points of the schematic (3).

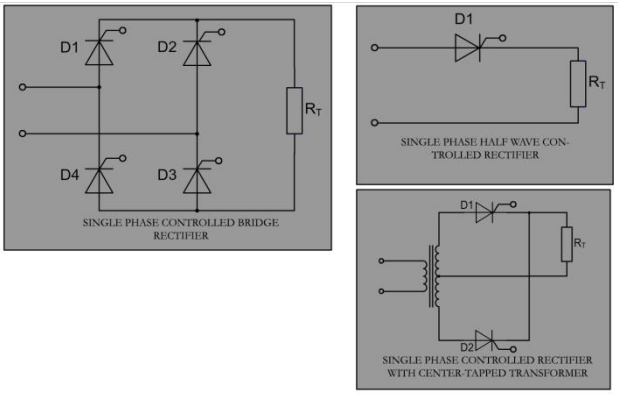


Fig. 1 Main view of the interactive application

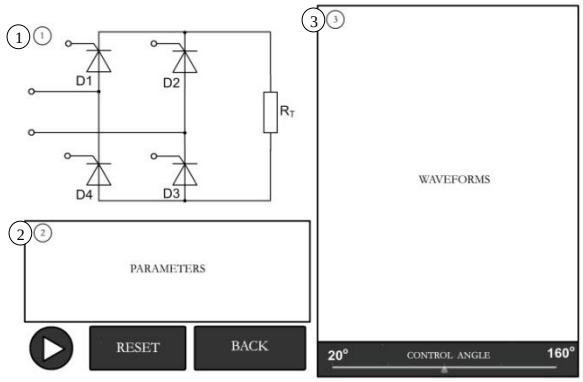


Fig. 2 Main view of single phase controlled bridge rectifier before simulation

The presentation is started by pressing an interactive button (fig.3). The graphics and parameters are color coded in order to be easily distinguished and associated with one another by the user. For example the voltage on Silicon Controlled Rectifier (SCR) D1 and SCR D3 is defined with blue. Thus it will be easily associated with the blue wave in the diagrams on the right and furthermore the resulted signals can be compared with one another.

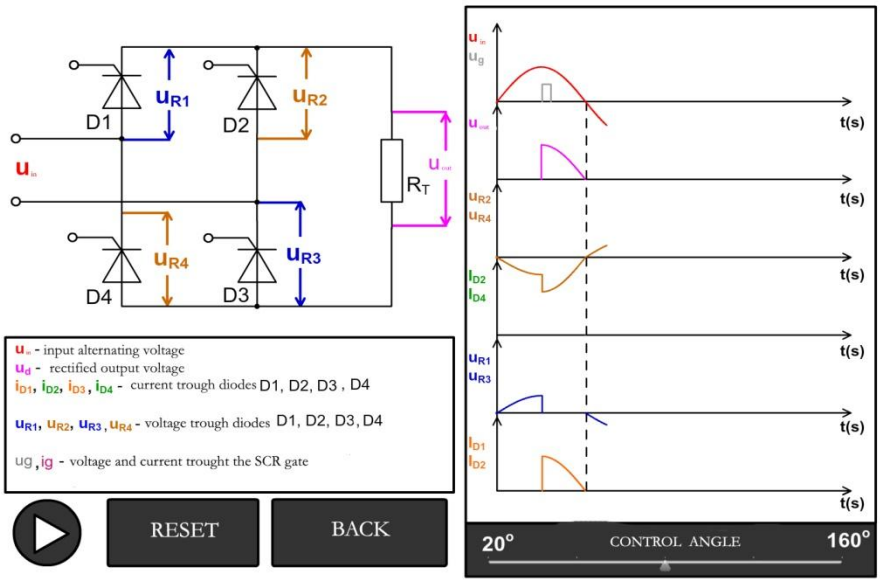


Fig. 3 Color coded example of running processes in the schematic

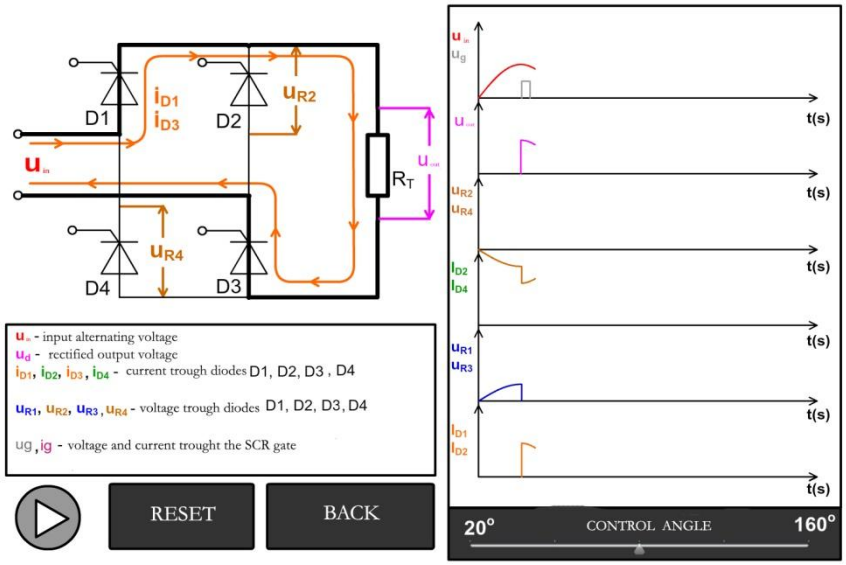


Fig. 4 Presents the current through D1 and D3

When a specific state of the components in the schematic is reached, this is easily discerned (fig.4 and fig.5). For example at first the current passes through SCRs D1 and D3 which is shown with orange on fig. 4. The green color defines the path of the current trough SCRs D2 and D4, which is also shown with the same color on the waveforms area on the right (fig. 5). Thus the change in state of the components, paths of current are clearly defined providing a scroll free screen and comprehensive definitions for the user. The different equivalent circuits are distinguished from the main topology through lines with higher width – combining in a single tab the main circuit and its equivalent branches.

An important parameter for the presented in the current paper schematic is the control angle. The control angle in this case can reconfigure the circuit and its parameters. Its effect however is difficult to explain through static images and waveforms. So in this case it is chosen that the control angle is to be defined by a slide control button. This is shown in fig. 6. Thus sliding the button to 20^o results in change of the control voltage on the SCRs, the current through them and consequently to a change of the output voltage of the rectifier. All this is implemented through a script file, implemented in the presentation where the theoretical formulas concerning the specified features

are defined. Thus an accurate and full description of the processes in current schematic allows the user to easily work and comprehend the main dependencies and work states without concerns.

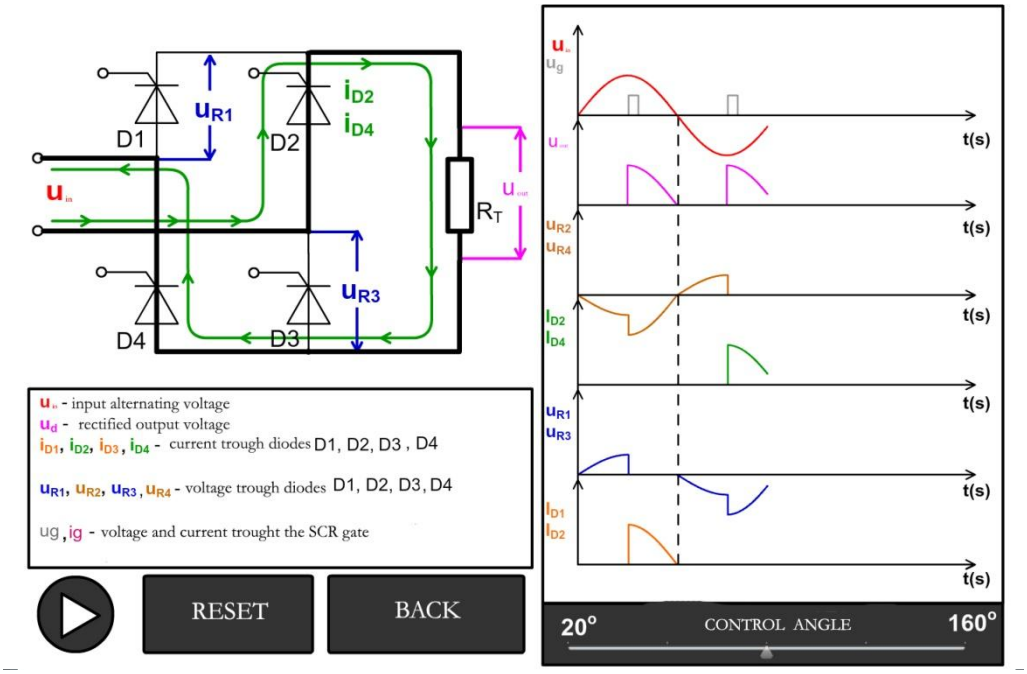


Fig. 5 Presents the current through D2 and D4

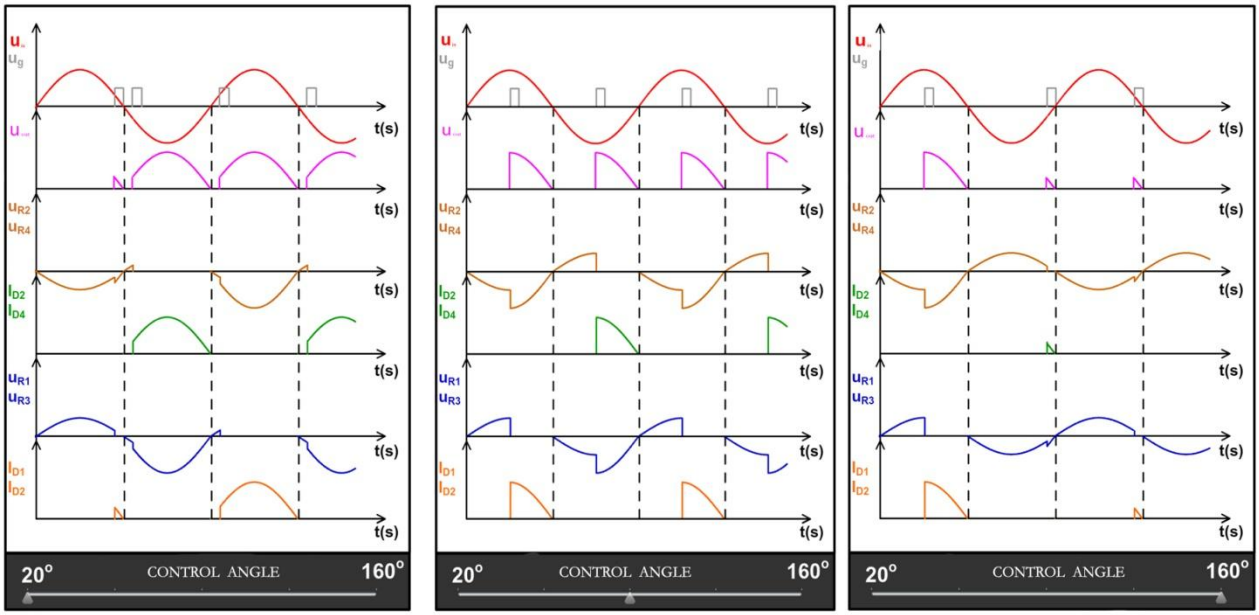


Fig. 6 Changing the control angle – from left to right waveforms for 20°, 90° and 160°

The presented example is developed in flash based software, where: the circuit and the text are developed through animation with multiple frames; the waveforms and the control buttons are developed through the utilization of action script 3.0. The example and other materials are currently being developed in HTML5 in order to increase the compatibility to additional hardware platforms – mobile phones, tablets, etc. The current example, its source, along with other specialized media for online education can be viewed at the web page of the Department of Electronics and Microelectronics of the Technical University of Varna (http://www.tu-varna.bg/tu-varnaetm/)

5. Conclusion

Based on a specialized literature review and experience gained through the development of two international projects three major issues related to the online education in power electronics were noted. The issues included difficulties in presenting complex information in a comprehensive way that will not distract the user and can keep his attention. Solution to deal with those issues were presented. The solution include the utilization of modern media techniques. An example of the presented solutions is included in the paper. The example along with other similar appliactions is available and can be used free of charge. It was also extensively used and tested in lectures of power electronics in the Technical University of Varna.

References

[1]. M.H.Rashid, "Power Electronics Circuits, Devices and Applications", 3rd Ed. Upper saddle, NJ: Pearson Prentice Hall. 2006.

[2]. Mohammed S., A. Moamen, B. Hasanin, A Review of the State-Of-The-Art of Power Electronics For Power System Applications, Quest Journals, Journal of Electronics and Communication Engineering Research, Vol. 1, Issue 1 (2013) pp: 43-52, ISSN(Online) :2321-5941

[3]. Класификатор на областите на висше образование и професионалните направления

[4]. Zhu J., Hybrid online-education strategy for delivering engineering and technology courses, 2nd International Conference on Networking and Digital Society (ICNDS), 2010, Volume:2, pages: 448 – 451; ISBN: 978-1-4244-5162-3

[5]. Drofenik, U., A. Musing; J. Kolar, Novel online simulator for education of power electronics and electrical engineering, International Power Electronics Conference (IPEC), 2010, Page(s): 1105 – 1111, ISBN: 978-1-4244-5394-8

[6]. Country report: Bulgaria, project: "BSUN Joint Master Degree Program on the Management of Renewable Energy Sources – ARGOS", 2011

[7]. V. Valchev (2013). „Interactive Multimedia Tools and Applications in Vocational Education in Wind Energy Technologies", International Jubilee Conference: 50-th Anniversary Department ETET, 2013, Varna, Bulgaria, 2013.

[8]. "Vocational Training in Wind Energy Technologies – Good Practices"; project "Transfer of Innovative VET System In Wind Energy Technologies" –TrainWIND, 2013

За контакти:
ас. д-р Ангел Станимиров Маринов
катедра „Електронна техника и микроелектроника"
Технически университет - Варна
E-mail: a.marinov@tu-varna.bg

ТЭХНАЛОГІЯ КІРАВАННЯ МАТЭРЫЯЛІЗАВАЦЬ ЎЯЎЛЕННЯМІ ЗАСНАВАНАЯ НА РАСПАРАДКУ РАБОТЫ АРГАНІЗАЦЫЙ

А.Б. Кунгурцев, Ю.М. Возовиков

Резюме: Тэхналогія кіравання матэрыялізаваць ўяўленнямі заснаваная на распарадку работы арганізацый. Прапануецца тэхналогія стварэння перыядычна падлучальных матэрыялізаваць уяўленняў (МП), заснаваная на распарадку працы арганізацыі. Гэты механізм прывязваючыся да гадзін працоўнага дня / днях тыдня / дэкадзе / месяца дазваляе павялічыць эфектыўнасць выкарыстання МП.
Ключови думи: інфармацыйныя сістэмы, матэрыялізаваць прадстаўлення, запыт.

MATERIALIZED VIEWS MANAGEMENT TECHNOLOGY BASED ON WORK SCHEDULE OF ORGANIZATIONS

A.B. Kungurtsev, U.N. Vozovikov

Abstract: Materialized views management technology based on work schedule of organizations. The technology of periodically connected materialized views (MV) forming on the basis of work schedule of organization is offered. This approach is linked to the working hours /days of week /decade /month and allows increasing of MVs implementation effectiveness.
Keywords: informational systems, materialized views, request.

Introduction

There are various software methods of increasing the productivity of informational systems (IS) based on using of relational data bases (RDB). One of them anticipates using of materialized views (MV) [1, 2, 3]. MV stores result of certain request to the data base (DB) and at following entry of this request to the IS allows getting answer very fast. However the practical use of MV in separate IS is impossible without preliminary study of this IS. As MV is a result of request which uses number of BD tables the refreshing of certain data in those tables leads to necessity of refreshing MV too. At frequent refreshing of data the using of MV could not increase but decrease the effectiveness of IS.

In a books [4, 5] the way of determining of those requests for which the using of MV will be effective on the basis of RBD requests consequence analysis is shown. AT this we consider that MVs in future will be connected permanently.

In this book the periodical connection and disconnection of MV is offered. The basis for this is an evident periodicity in resolving of different tasks for the most of organizations. For example, there is a period of admission, passing the tests, and visiting hours in the university. In trading companies the periodicity of goods income, residues inventories, revaluation and sales out, seasonal variations in assortment and working hours is observed. The periodicity of resolving of production objectives is reflected in periodicity of requests incoming to the system.

Advantages of MV management

Periodical connection/disconnection of MV will allow to increase its effectiveness due to following factors.

1. Possibility of disconnection of some MV during period when it's using is non-effective.

2. Possibility of initiation into service of MVs which became non-effective at continuous connection but which are effective during certain periods.

3. Lowering of DBMS resources spent for servicing of MV mechanism due to implementation of effective MVs at now only.

Reasoning about efficiency of MV

Let us assume that *Q* is a consequence of requests being sent to the IS during period of observation t_0. There are n_i requests of q_i form and of SELECT type in this consequence. To determine the possibility of forming MV for requests q_i, introduce the term «effectiveness *Mpi*» (materialized view for request q_i), specified as relation of all q_i requests execution time without implementation of MV to the time of all q_i requests execution time with implementation of MV.

$$Ec_i = \frac{S0_i}{Smp_i + Snew_i + Ssel}, \quad (1)$$

where: $S0_i = \sum_{j=1}^{n_i} t_i$ — total time of all q_i requests execution during observation period t_0 without implementation of MV;

$Smp_i = n_i * t_{mpi}$ — total time of all q_i requests execution with implementation of *MPi*;

$Snew_i = ku_i * tu_i$ — time of *MPi* refreshes during t_0. Here ku_i is a quantity of *MPi* refreshes which is determined by number of UPDATE, INSERT and DELETE requests which changes the data in basic tables affecting the value of *MPi*.

tu_i - average time of refresh *MPi*.

$Ssel = n * n_{mp} * t_s$ - time being spent for choosing the requests having MV from common flow of requests. Here n is a quantity of all requests came into DB during time of observation t_0, n_{mp} is a quantity of MVs being used.

Let us introduce the term of effectiveness of managed implementation of MV. At this we will call the time interval during which *Mpi* is connected for the time $\tau 1_i$ (connection period) and then is disconnected for the time $\tau 0_i$ (disconnection period) as management period.

$$E\tau_i = \frac{S0_i}{Smp\tau_i + Snew\tau_i + Ssel\tau + Sbd}, \quad (2)$$

where: $Smp\tau_i$ is a total time for all q_i requests execution which are in periods $\tau 1_i$.

$Snew\tau_i$ is a time of *MPi* refreshes during $\tau 1_i$ periods.

$Ssel\tau = n * n\tau_{mp} * t_s$ is a time being spent for choosing the requests having MV from common flow of requests. Here $n\tau_{mp}$ is a quantity of MVs being implemented taking into account the introduction of connection/disconnection mode ($n\tau_{mp} < n_{mp}$).

$Sbd = (n_i - n\tau_i) * t_i$ is a time being spent for q_i requests execution which are not in $\tau 1_i$ periods.

Let us introduce the term of maximum effectiveness of *Mpi* implementation calculated at condition that only those periods of observation are taken into account in it for which $E\tau_{ij}>1$. Averaged value of such effectiveness's will be considered as maximum effectiveness of *Mpi* implementation.

$$E\tau\max_i = \sum_{j=1}^{m} E\tau_{ij} / m, \tag{3}$$

Choosing of way of MV implementation

Value $E\tau\max_i$ plays role of upper limit of possible *Mpi* effectiveness and is a component of dependency of MV implementation effectiveness form the method of MV management specified in following rules.

1. If for some request q_i $Ec_i > 1$ and $E\tau\max_i > Ec_i$ then it is reasonable to search the periods of connection/disconnection of *Mpi.*
2. If for some request q_i $Ec_i > 1$ and $Ec_i \approx E\tau\max_i$ then it is no need to search the periods of connection/disconnection of *Mpi,* but need to connect the *Mpi* for all period of AS operation.
3. If for some request q_i $Ec_i < 1$ and $E\tau\max_i > Ec_i$ then it is reasonable to search the periods of connection/disconnection of *Mpi.*
4. If for some request q_i $Ec_i < 1$ and $E\tau\max_i < Ec_i$ then no sense in *Mpi* implementation.

Determining the parameters of MV management

Let`s consider as it is possible to find several effective periods of connection/disconnection for *MPi* where each of periods is specified by duration of connected and disconnected condition of *MPi*

$\tau_{ij} = \tau 1_{ij} + \tau 0_{ij}$.

For estimation of effectiveness of each certain *MPi* connection/disconnection period implementation the formula (4) is offered where numerator is determined by not all requests q_i but only those which are in connection periods *Mpi*. This allows assessment of effectiveness for each period having been found by separate.

$$E\tau_{ij} = \frac{S0\tau_{ij}}{Smp\tau_{ij} + Snew\tau_{ij} + Ssel\tau} \tag{4}$$

Here $S0\tau_{ij} = \sum_{r=1}^{n\tau_i} t_i$ is a total time of all q_i requests execution during periods $\tau 1_{ij}$ without implementation of MV.

Relative assessment of some management period effectiveness $E\tau_{ij}$ is necessary for assessment of this period but not allow estimate the «deposit» of this period to total effectiveness *Mpi*. To estimate the «deposit» of each management period and determinate the end of new management periods search process let us to introduce the absolute estimates of effectiveness specified by reducing the time for requests execution.

In accordance with (1) we will obtain reducing of time at constant connection of *Mpi* - Δtc_i.

In accordance with (3) we will obtain maximum reducing of time at selective connection *Mpi*- $\Delta t\max_i$.

In accordance with (4) we will obtain reducing of time at choice of certain management period *Mpi*- $\Delta t\tau_{ij}$.

The basis of management periods and corresponding connection $\tau 1_{ij}$ and disconnection $\tau 0_{ij}$ periods search method is a repetitive process which ends when value of total reducing of time at periodical management *Mpi*

$$\Delta t_i = \sum_j^m \Delta t\tau_{ij} \cong \Delta t \max_i, \quad (5)$$

or there is no possibility for addition of new management periods, for example, increasing *m* is impossible for existing value of observation time t_0.

Determination of management periods on the basis of work schedule of organization

In real informational system the periods of MV management could be linked with work schedule of organization that is to the working hours, day of week, date of month and so on. In addition, depending on particular characteristics of organization profile the specialized methods could be introduced, for example, 3 days before each month end, 5 days before each quarter end etc.

On the basis of IS operation analysis and convenience of MV management it is convenient to choose one hour as the shortest period of connection. Let us to introduce the numbering of IS working hours from the moment of beginning of working day.

Based on the above the first period of management is offered to be set the week and period of connected state *Mpi* is a first hour of Monday. Then with saving of management period we set the connection period *Mpi* as second hour of Monday and so on. While process continue all hours of each day of the week are processed consequently with saving the period which is equal one week.

For each day an hour on the basis of (4) the effectiveness is calculated

$$E\tau_{iWeek,Dj,Hk} = (\sum_1^n E\tau_{iDj,Hk}) / n,$$

where *n* is a quantity of weeks entered in observation period,

Dj is a day of week,

Hk is a number of hour of working day.

The days and hours used for calculation of three values of effectiveness $E\tau_{iWeek,Monday,1}$ $E\tau_{iWeek,Tuesday,3}$ $E\tau_{iWeek,wednesday,4}$ are specified in table 1.

Table 1. Management period is a week

Hour	Week							Week							Week…		
	Mn	Tu	Wd	Th	Fr	Sa	Sn	Mn	Tu	Wd	Th	Fr	Sa	Sn	Mn	Tu	Wd
1	■							■							■		
2																	
3		■							■							■	
4			■							■							■
5																	
6																	
7																	
8																	

The results of calculations are convenient to be represent in view of matrix of management *Mpi*, where each cell has 0 (corresponds to value $E\tau_{iWeek,D,H} \le 1$) or 1 (corresponds to value $E\tau_{iWeek,D,H} > 1$)

Table 2. Management of *Mpi* on the basis of weekly analysis

Hours	Days of week						
	Mn	Tu	Wd	Th	Fr	Sa	Sn
1	1	0	1	0	1	0	0
2	1	1	1	0	0	1	0
3	0	0	1	1	0	0	0
4	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0
6	1	0	1	0	0	0	0
7	1	0	1	0	0	0	0
8	1	0	0	1	0	0	0

There is a conclusion from table 2 particularly which *MPi* could be connected in Monday during 1st and 2nd hour as well as during 6, 7 and 8 hour.

Introduction of new more long periods of management allows correction of obtained tables of management *MPi*. Consider how to take into account possible periodicity of requests concerned with decades of month.

If earlier the averaging of effectiveness by weeks were performed independently of arrangement of certain week in month (first or last) then now averaging is performed by each of decades. For each of decades, day of week and hour the effectiveness is calculated

$$E\tau_{iTen_l,Dj,Hk} = (\sum_{1}^{n} E\tau_{iDj,Hk})/n,$$

where *n* is a quantity of days *Dj* entered in period of observation,

l – number of decade.

The results of decades analysis are shown in table 3 fragmentally

Table 3. Management *Mpi* on the basis of analysis by decades

Hours	Decade 1				Decade 2				Decade 3			
	….	Mn	Tu	….	….	Mn	Tu	….	….	Mn	Tu	….
1		1	0			1	0			1	1	
2		1	0			1	0			1	1	
3		0	0			0	0			1	0	
4		0	0			0	0			0	0	
5		0	0			0	0			1	0	
6		1	0			1	0			1	0	
7		1	0			1	0			1	0	
8		1	0			1	0			1	0	

The correcting of management *MPi* taking into consideration of study of more long-time management period.

Limitation for duration of management period being analyzed is made by observation time t_0 for operation of AS. It is recommended to limit the duration of longest management period by value of $t_0/2$.

This assessment together with (5) is a condition of new management periods search process end.

The duration of management period could be increased according to the results of observation for IS during time of system maintenance at condition of continue of requests consequence studying.

Conclusion

The analysis of requests consequence came into automation system having been performed had allow determine the requests which are profitable to connect at constant basis and other requests should be connected periodically. Total time of requests execution was decreased. MVs being connected periodically in period of intense reports preparation allows avoiding of peak loads on the system.

Developed method of MV management could be used in many organizations where the certain periodicity of tasks being executed is observed.

Literature

[1]. Advenced Information Systems. Oracle 8. User cyclopedia: tr. from eng./ Advenced Information Systems // «DiaSoft» publishing. Kyiv, 1998, — 864p.

[2]. Robert Viera. Programming of Microsoft SQL Server 2005 data bases. Basics.: tr. from eng. / Robert Viera // «Dialektica» publishing. Moscow, 2007, — 832p.

[3]. Alexey Vishnevsky. Microsoft SQL Server. Efficient work./ Alexey Vishnevsky // «Piter Press» LLC. Piter, 2009, — 541p.

[4]. Kungurtsev A.B. Search of regularities in distribution of requests for management of materialized views/ Kungurtsev A.B., Vozovikov U.N.// Odessa National Polytechnic University. Odessa, 2008. — 2(30). — p. 135— 140.

[5]. Kungurtsev A.B. Analysis of possibility of MV implementation in IS/ A.B. Kungurtsev, Kuok Vin Nguen Chan // Odessa National Polytechnic University. Odessa, 2004. — 2(20). — p. 102-106.

[6]. Kungurtsev A.B., Management of materialized views in nformational systems/ Kungurtsev A.B., Vozovilov U.N. // East-European magazine of modern technologies. Kharkov, 2010. — ¼(43). — p. 18 — 21.

For contacts:
Academic, PhD in Technical Sciences, A.B. Kungurtsev,
Ukraine, Odessa`s National Polytechnic University,
E-mail: abkun@te.net.ua

Scientist, U.N. Vozovikov,
Ukraine, Odessa`s National Polytechnic University,
E-mail: yuri_email@mail.ru

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

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

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

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

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

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

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

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