В таблиці наведені результати дослідження ефективності описаної методики використання узгоджених кодів N'_j для генерації 12–розрядним ЦАП гамонічних сигналів частотою 50 Гц (Δt=312,5 мкс). При цьому кількість точок дискретизації N = 64. Для порівняння наведені відносні похибки формування діючого значення $\delta \dot{U}'$ і значень коефіцієнтів амплітуди $\delta k'_a$ та форми $\delta k'_f$ синусоподібних сигналів, які сформовані на основі кодів миттєвих значень N_j, отриманих згідно з виразом (1), і узгоджених кодів N'_j, обчислення яких виконувалось за співвідношеннями (11) та (12). При цьому розрахункові величини даних параметрів такі: діюче значення напруги синусоподібного сигналу $\dot{U}$ =2895,6 (В); коефіцієнт амплітуди k_a = 1,14142; коефіцієнт форми k_f = 1,11072.

Аналізуючи наведені дані, можна зробити такі висновки:

– значення параметрів вихідних каскадів генераторів гармонічних сигналів мають істотний вплив на якість формування вихідних сигналів. Внаслідок інерційності ЦАП виникають нелінійні спотворення, які впливають не тільки на формування діючого значення гармонічних сигналів, а також і на їх форму;

– для покращання інтерполяції вихідних сигналів між їх точками дискретизації і, відповідно, усунення нелінійних спотворень, не обов'язково зменшувати крок їх дискретизації, що виконувалось традиційно, а достатньо виконати узгодження кодів миттєвих значень згідно з (11) та (12);

– додаткові дослідження показали, що оптимальне значення співвідношення між кроком дискретизації Δt та постійною часу інтегруючої ланки τ лежить в інтервалі від 1,5 до 2 і залежить від розрядності *e* ЦАП, який на основі кодів миттєвих значень виконує формування вихідного сигналу;

– при заданій продуктивності процесора або мікро-ЕОМ, на основі яких розробляється цифровий генератор, за рахунок введення додаткової RC-ланки на виході ЦАП і узгодження кодів миттєвих значень можна значно збільшити частоту дискретизації вихідних сигналів і, за рахунок цього, спростити внутрішню структуру і розширити сервісні можливості пристрою.

УДК 681.3

Почаєвець О.М.
ДУ "Львівська політехніка", кафедра ЕОМ

БІБЛІОТЕКА ФУНКЦІЙ ШВИДКОГО ОПРАЦЮВАННЯ ДАНИХ РЯДКОВОГО ТИПУ

Запропоновано швидку реалізацію стандартних функцій опрацювання даних рядкового типу, згрупованих в бібліотеку з метою їх використання в програмах, написаних мовою С.

Вступ. Стандарт мови ANSI C передбачає використання набору рядкових функцій після підключення директивою include їх опису з файлу string.h до програми користувача: #include < string.h>. Пропоновані стандартом ANSI рядкові функції мають таку низку особливостей [1].

1. Вхідні/вихідні параметри рядкових функцій є вказівниками на рядок. Це передбачає додатковий сторонній розподіл пам'яті для рядків, що опрацьовуються, а також обмежує вкладеність цих функцій одна в одну при побудові виразів обчислень рядків.

2. Форма представлення рядка набором символів, що закінчуються символом '\0', вимагає частого підрахунку його довжини. Це збільшує термін опрацювання рядків.

3. Рядки опрацьовуються побайтово, хоча в сучасних 32- та 64-бітових ОС можна було б прискорити їх опрацювання, задіявши повну розрядну сітку комп'ютера.

Реалізація. Реалізовано бібліотеку функцій швидкого опрацювання даних рядкового типу за такими принципами:

1. Бібліотека містить стандартні рядкові функції ANSI C [1] як базові, а також має низку додаткових функцій роботи з рядками, формат яких запозичений з мови програмування Clipper Summer'87 [2]. Необов'язкове використання додаткових функцій дозволяє опрацьовувати рядки більш гнучко. Перелік додаткових рядкових функцій наведено на рис.1.

1) notempty	15) rightl	29) padc
2) len	16) substr	30) strraw
3) at	17) strtran	31) strunraw $$ strdump
4) rat	18) ltrim	32) string_time
5) cmp && cmp_s	19) rtrim	33) strings_version
6) equ	20) alltrim	34) sch2str
7) equ_(f) num	21) pack	35) ssh2str
8) cat	22) head	36) slo2str && ptr2str
9) cat1	23) tail	37) ldf2str
10) space	24) upper	38) str2sch
11) replicate	25) lower	39) str2ssh
12) left	26) rev	40) str2slo && str2ptr
13) leftr	27) padl	41) str2ldf
14) right	28) padr	42) free string

Рис. 1. Перелік додаткових рядкових функцій

2. Вхідні параметри рядкових функцій передаються у вигляді вказівників (char*), а вихідний – у вигляді вказівника на вказівник (char**). Такий підхід дозволяє динамічно виділяти пам'ять під результуючий рядок, як показано на рис.2

```
char* func (char **target, char *source) {
  char *targ=*target;
  … /* count newsize */
  targ=(char*)realloc((void*)targ, newsize);
  *target=targ;
  return targ;
}
```

Рис.2. Загальна внутрішня структура рядкової функції

Отже, з рис.2 видно, що за допомогою вказівника на вказівник target, нова адреса результату передається програмі, з якої було здійснено виклик функції. Додаткове повернення вказівника на результуючий рядок targ надає можливість побудови виразів опрацювання рядків шляхом вкладення функцій одна в одну. Приклад:

```
char  *target, *source;
…
func1 (&target, func2(&target, source));
```

3. В реалізації функції використовується Холеритівський принцип представлення рядків у пам'яті [3] за модифікованою схемою, наведеною на рис.3.

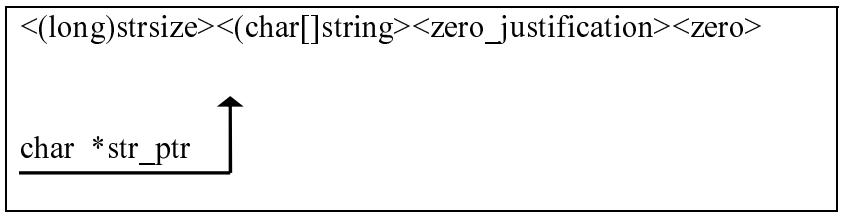

Рис.3. Формат представлення рядків у пам'яті

Рядок містить власну довжину strsize, але вказівник на рядок вказує не на довжину, а безпосередньо на початок рядка. Залежно від розміру розрядної сітки комп'ютера кінець рядка доповнюється символами '\0' zero_justification, що дає змогу опрацювати його не побайтово, а за повними словами розрядності комп'ютера. Додаткове значення zero в кінці рядка вводиться для сумісності модифікованого формату із стандартними функціями ANSI C, що працюють в звичайному форматі представлення рядків [1]. Холеритівське запам'ятовування довжини рядка прискорює опрацювання рядків майже в два рази за рахунок вилучення підрахунків довжин рядків в процесі опрацювання. Додатково надається можливість включення символу '\0' до складу символів рядка.

4. Запропонований модифікований Холеритівський формат рядків дозволяє опрацювати рядки з використанням повної розрядної сітки комп'ютера, як показано на рис.4

```
l=len(source)-len(source) &sizeof(long);
for(i=0; i<l; i+=sizeof(long))
  *((long*)target+i)=*((long*)source+i);
for(; i<len(source); i++)
  *(target+i) =* (source+i);
```

Рис.4. Копіювання рядка source в рядок target
з використанням повної розрядної сітки комп'ютера

Для 32-розрядник комп'ютерів запропонований підхід опрацювання рядків дає прискорення в часі майже в чотири рази у порівнянні із стандартним побайтовим опрацюванням.

Висновки. Запропоновано та реалізовано бібліотеку функцій швидкого опрацювання даних рядкового типу. Для 32-розрядних комп'ютерів виграш у швидкодії наближено становить вісім разів порівняно із швидкодією стандартних ANSI C функцій. Бібліотека крім базових функцій містить низку додаткових та не вимагає стороннього розподілу пам'яті під рядки, що опрацьовуються. Разом з можливістю гнучкої побудови рядкових виразів запропонована бібліотека є швидким та потужним засобом для побудови програм, що займаються опрацюванням рядкової інформації.

Запропонована бібліотека реально використана при побудові синтаксичного аналізатора в системі автоматизованого розпаралелювання та виконання однопотокових послідовних програм [4].

1. ANSI C: ANSI Technical Committee X3J11, 1988. 2. Clipper Summer'87 Programmer's Guide. 3. Грунд Ф. Программирование на языке ФОРТРАН IV, М., 1974. 4. Почаєвець О., Система автоматизованого розпаралелювання та виконання однопотокових послідовних програм / Вісн. ДУ "Львівська політехніка".

УДК 62.519

Рицар Б.Є.
ДУ "Львівська політехніка", кафедра РТП

ПРО ДЕКОМПОЗИЦІЙНІ КЛОНИ БУЛЕВИХ ФУНКЦІЙ

Розглядається теоретико-множинний критерій декомпоновності булевих функцій *n* змінних, що ґрунтується на запропонованому понятті так званого декомпозиційного клона. Показано, як шляхом простих процедур розв'язується проблема доозначення часткових булевих функцій. Формулюється теорема про роздільну декомпозицію булевих функцій *n* змінних.

Декомпозиція, як процес вираження логікової функції n змінних за допомогою функцій меншої кількості змінних, є фундаментальною проблемою логікового синтезу [1,2]. Зокрема, булева функція $Y = f(X) = f(x_1, x_2, ..., x_n)$ вважається декомпоновною, якщо її можна виразити одним з трьох видів:

$$Y = \varphi(\varphi_1(X^{n-q}), X^q); \quad (1)$$

$$Y = \varphi(X^{n-q}, \varphi_1(X^q)); \quad (2)$$

$$Y = \varphi(\varphi_1(X^{n-q}), \varphi_2(X^q)) \quad (3)$$

де X^{n-q}, X^q – власні підмножини множини змінних $X = \{x_1, x_2, ..., x_n\}$, утворені процедурою q-розбиття мінтермів [3,4], причому, у разі роздільної декомпозиції $X^{n-q} \cap X^q = \varnothing$; $q \in \{1,2,...,(n-1)\}$; φ_1, φ_2 і φ – відповідно, зв'язані і залишкова булеві функції меншої кількості змінних, ніж задана функція Y.

Класичним критерієм декомпоновності булевої функції Y, тобто здатності бути розділеною до виду одноблокової (1,2) чи двоблокової (3) декомпозиції, є не більше двох різних рядків чи/і стовців декомпозиційної карти чи карти Карно [1,5] або таблиці істинності [6] чи булевої матриці [7].

У даній статті пропонується новий критерій функційної декомпоновності, що ґрунтується на процедурі q-розбиття мінтермів функції Y, заданої теоретико-множинною формою (ТМФ) [4]. Зокрема, повна ($p = 0$) або часткова (недоозначена чи слабоокреслена)