Значення кодів N_1 , N_2 та N_3 визначаються перед початком аналого-цифрового перетворення поточного сигналу з множини вхідних сигналів $\{U_I\}$ і для зменшення впливу випадкових похибок можуть усереднюватись.

Враховуючи, що, в основному, АЦП використовуються для контролю періодичних сигналів, то попередню оцінку вхідного сигналу можна виконувати один раз на початку періоду його повторення, а необхідне зміщення для перетворення наступної різниці визначати програмним шляхом на основі аналізу старших розрядів попередньої різниці. Якщо ж визначати початок періоду в момент часу, коли миттєве значення вхідного сигналу наперед відоме (наприклад, приблизно дорівнює нулю), то можна відмовитись від його попереднього оцінювання. У кожному випадку, аналого-цифрове перетворення миттевих значеннь сигналів, що контролюються, можна виконувати за один такт (без врахування додаткових тестуючих перетворень).

Висновки. Розглянуті в статті особливості виконання диференціального аналого-цифрового перетворення дозволяють розробити високоточний і швидкодіючий АЦП на основі сучасних малорозрядних інтегральних мікросхем. Приведений алгоритм поточного визначення зміщення нуля і коефіцієнта передачі аналого-цифрового перетворення дозволяе автоматично корегувати адитивну та мультиплікативну систематичні похибки АЦП і підвищує стабільність функціонування інформаційно-вимірювального пристрою при дестабілізуючих впливах.

1. ГОСТ 26035-83 Счётчики электрической энергии переменного тока электронные. 2. Орнатский П.П. Автоматические измерения и приборы. – К., 1973. 3. Бахтиаров Г.Д., Малинин В.В., Школин В.П. Аналого-цировые преобразователи / Под ред. Г.Д. Бахтиарова. – М., 1980. 4. Дороніна О.М., Лавров Г.М., Паньків Р.С, Хомич С.В. Аналого-цифровий перетворювач з програмованим діапазоном вхідного сигналу // Вісн. Держ. ун-ту "Львівська політехніка". – 1998. – № 350. – С. 115–117.5. Обозовський С.С. Інформаційно-вимірювальна техніка // Методологічні питання теорії вимірювань. – К.: ІСДО, 1993. 6. Цветков Э.И. Процесорные измерительные средства. – Л.: Энергоатомиздат, 1982. 7. Бродин В.Б., Калинин А.В. Системы на микроконтроллерах и БИС программируемой логики. – М.: Издательство ЭКОМ, 2002.

УДК 681.322

Р.Б. Попович
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ПРО ОТРИМАННЯ ВЕЛИКИХ ПРОСТИХ ЧИСЕЛ

© Попович Р.Б., 2004

Розглядається шлях генерації випадкових великих простих чисел.

A way of generation of random big prime numbers is considered.

Вступ. Багато криптосистем використовують параметр, який є простим числом [1]. У цьому разі просте число тримається в таємниці від зловмисника. Щоб позбавити зловмисника шансів зламати криптосистему, це просте число треба вибирати випадково і воно має бути великим.

Вибір простих чисел гарантується в широкому асортименті оцінкою Чебишова та низкою пізніших результатів про розподіл простих чисел [2]. Метою цієї роботи є спрощення процедури отримання великих простих чисел, що призведе до зменшення часу, необхідного для виконання цієї процедури.

Огляд методів генерування випадкових простих чисел. Утворення випадкового простого числа заданого розміру (із заданою кількістю десяткових розрядів *к*) рекомендують виконувати так

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

[1]. Вибирається випадкове натуральне число n між $10^{\kappa-1}$ і 10^{κ}. Далі по черзі перебирають числа n, n+1, n+2 , кожне з яких випробовується тестом простоти. Ці випробування триває доти, поки якесь число n+m не витримає тест. Воно й береться як шукане просте число.

Зупинимось на питанні, якого часу вимагає описана процедура. Зрозуміло, що він обмежується величиною O(m t), де n+m – перше просте число після m, а t – час роботи тесту простоти, що використовується. Хоча справедливість гіпотези $m(n)$=(log n)O(1) невідома, наступний евристичний вираз не розходиться з емпіричним досвідом [1]. Асимптотичний закон розподілу простих чисел дає підстави сподіватися, що від n найближче просте число знаходиться на відстані щонайбільше ln n=k ln 10.

Відомі сьогодні тести простоти діляться на два класи: детерміновані та ймовірнісні тести. Час виконання тестів з другого класу значно менший, ніж тестів з першого класу.

Існують різні ймовірнісні тести простоти чисел, які визначають, чи є число простим, із заданим степенем достовірності. За умови, що цей степінь достовірності досить великий, такі способи перевірки є досить добрі. Прості числа, згенеровані таким способом, називають промисловими або комерційними простими числами: ці числа ймовірно є простими з контрольованою можливістю помилки.

Найвідомішими на сьогодні ймовірнісними тестами простоти є тест Соловея–Штрассена (та його модифікація Лемана) і тест Рабіна–Міллера [1–3].

Для перевірки простоти числа p вказаний тест Соловея–Штрассена використовує поняття символа Якобі [1] і зводиться до послідовності таких кроків:

1. Вибрати випадкове число a, менше за p.
2. Якщо найбільший спільний дільник a та p не дорівнює 1, то p – складене число.
3. Обчислити j=$a^{(p-1)/2}$ mod p.
4. Обчислити символ Якобі J(a,p).
5. Якщо $j\neq$ J(a,p), то число p складене.
6. Якщо j= J(a,p), то ймовірність того, що число p складене, не перевищує 1/2.

Коли повторити перевірку t разів з t різними значеннями a, то ймовірність того, що складене число пройде всі t перевірок, не перевищує $1/2^t$.

Модифікація Лемана полягає в тому, що символ Якобі не обчислюється, а величина $a^{(p-1)/2}$ mod p просто порівнюється з 1 або –1. Імовірність того, що складене число пройде t перевірок, також не перевищує $1/2^t$.

Коли повторити перевірку t разів з t різними значеннями a, то ймовірність того, що складене число пройде всі t перевірок, не перевищує $1/2^t$.

Модифікація Лемана полягає в тому, що символ Якобі не обчислюється, а величина $a^{(p-1)/2}$ mod p просто порівнюється з 1 або –1. Імовірність того, що складене число пройде t перевірок, також не перевищує $1/2^t$.

У тесті Рабіна–Міллера ймовірність проходження перевірки складеним числом спадає скоріше, ніж у двох попередніх тестах, та не більша $1/4^t$.

Тест Рабіна–Міллера зводиться до таких дій. Число p, що перевіряється, спочатку зображаємо у вигляді p=2^b m+1, де m – непарне. Для випадкового числа a, меншого за p та взаємно простого з ним, обчислюємо x_0=a^m mod p. Якщо x_0=±1 mod p, то p просте. В іншому разі обчислюємо x_i=(x_{i-1})^2 mod p (m=1,2,...,b). Коли на якомусь кроці маємо -1, то p просте, а коли ні – p складене.

Основа алгоритму. Думка полягає в тому, щоб не тестувати на простоту, як загально прийнято [1], всі послідовні натуральні числа, починаючи з деякого випадково вибраного натурального числа. Адже кожен тест на простоту вимагає значних затрат часу [1, 2]. Замість цього можна спочатку вибрати такий випадковий інтервал чисел [n,n+ln n], на якому є мінімальна кількість кандидатів на прості числа, а лише потім виконувати для цих відібраних чисел необхідні тести простоти.

Утворення випадкових великих простих чисел. Сформулюємо послідовність дій, необхідних для отримання випадкового великого простого числа.

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

Спочатку генеруємо випадкове натуральне число n із заданою кількістю $к$ десяткових розрядів. Зауважимо, що замість кількості десяткових розрядів можна задавати й кількість двійкових розрядів (біт).

Далі відбираємо ті натуральні числа з інтервалу $[n,n+k \ln 10]$, які можуть бути простими. Для цього послідовно знаходимо залишки від ділення $n+m$, m =0, 1, …,k ln 10 на наперед обчислені невеликі прості числа, менші від деякого наперед вибраного натурального числа. Тут це число дорівнювало 30000. Кількість простих чисел, менших від 30000, дорівнює 3245. Тоді для кожного числа $n+m$ обчислюємо 3245 залишків. Безпосередньо обчислюємо цей набір залишків тільки для першого числа n з вказаного інтервалу. Для всіх решта чисел $n+m$ залишки отримуємо додаванням одиниці до відповідних залишків для попереднього числа $n+m$-1. Додавання виконуються за модулем відповідних простих чисел p_i, i=1,2,…,3245.

Підраховуємо, скільки є кандидатів на прості числа на інтервалі $[n, n+k \ln 10]$. Якщо кількість кандидатів виявляється меншим від отриманого експериментально мінімальної кількості кандидатів для цього числа десяткових розрядів $к$, то генеруємо нове випадкове натуральне число n із заданою кількістю $к$ десяткових розрядів і повторюємо описані дії.

Вказані дії реалізуються у такому алгоритмі.

Алгоритм 1. Відбирання чисел, які можуть бути простими

Вхідні дані: $к$ – потрібна кількість десяткових розрядів простого числа;
lmin – отримана експериментально мінімальна кількість кандидатів для даного числа десяткових розрядів $к$

Крок 1. *limit*:= round(k ln 10) – довжина інтервалу, на якому згідно з оцінками про розподіл простих чисел шукаємо просте число

Крок 2. Генеруємо випадкове $к$-розрядне десяткове число n. Його молодший розряд має бути непарним і відмінним від 5, а старший розряд – відмінним від нуля

Крок 3. m:=0
l:=0

Крок 4. Знайти залишки від ділення $n+m$ на наперед обчислені прості числа p_i, менші 30000 (як знаходити залишок від ділення числа на невелике просте число див. алгоритм 2)

Крок 5. Якщо хоча б один отриманий на кроці 4 залишок дорівнює нулю, то перейти на крок 7

Крок 6. $d[l]$:=m
l:=l+1

Крок 7. m:=m+1

Крок 8. Якщо m перевищує *limit*, то перейти на крок 11

Крок 9. Всі залишки збільшити на одиницю за відповідними модулями p_i

Крок 10. Перейти на крок 5

Крок 11. Якщо l дорівнює *lmin*, то перейти на крок 12. У протилежному випадку, перейти на крок 2

Крок 12. Зберегти масив $d[l]$, l=1,…,*lmin*

Вихідні дані: $n+d[l]$, l=1,…, *lmin* – послідовність натуральних чисел з інтервалу $[n, n+\text{round}(k \ln 10)]$, які можуть бути простими

Зауважимо, що перед виконанням наведеного алгоритму для заданого числа $к$ десяткових розрядів *prob* разів здійснювався вибір інтервалу натуральних чисел довжини round(k ln 10) та підрахунок кількості кандидатів на прості числа на цьому інтервалі . Тут число *prob* дорівнювало 1000. У результаті знайдено мінімальні числа *lmin* кандидатів на прості числа для різних $к$.

Алгоритм 2. Знаходження залишку багаторозрядного числа *a* за модулем невеликого простого числа *p*

Вхідні дані: $a=a_n\ a_{n-1}\ \ldots a_0$ –число в системі числення за основою *osnova*=10,
p – невелике просте число

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

Крок 1. res:=0
w:=1
i:=0
Крок 2. $dres$:=$w \cdot a_i$ mod p
Крок 3.res:=(res+$dres$) mod p
Крок 4. w:=$osnova \cdot w$ mod p
Крок 5. i:=i+1
Крок 6. Якщо $i \le n$, то перейти на крок 2
Вихідні дані: залишок a mod p дорівнює res

Зауважимо, що основу числення *osnova* не обов'язково брати такою, яка дорівнює 10 – вона може бути довільною.

Після того, як кандидатів на прості числа відібрано, для кожного з них виконуємо якийсь з описаних раніше тестів простоти. Скажімо, це може бути спрощений варіант модифікації Лемана, коли обчислюється величина $2^{(p-1)/2}$ mod p та порівнюється з 1 і –1.

Оцінка обчислювальних затрат. Виконано експерименти для пошуку простих чисел з різним великим числом десяткових розрядів ($к$ = 50, 100, 150, 200). Для цього написана програма на мові Object Pascal в середовищі візуального програмування Delphi 7.0. Отримані для різної розрядності $к$ простих чисел результати експериментів наведені в таблиці.

Кількість розрядів $к$	Довжина інтервалу	Середня кількість кандидатів на прості числа	Мінімальна кількість кандидатів на прості числа
50	115	6	2
100	230	13	5
150	345	19	8
200	460	25	14
250	575	32	18

Висновки. Запропонований підхід дає змогу зменшити обсяг необхідних обчислень для отримання багаторозрядних (50–200 розрядів) простих чисел приблизно в 1,8 раза.

1. Вербіцький О. В. Вступ до криптології. – Львів, 1998. 2. Ємець В.Ф., Мельник А.О., Попович Р.Б. Сучасна криптографія. Основні поняття. – Львів, 2003. 3. Шнайер Б. Прикладная криптография. Протоколи, алгоритми, исхолние тексти на язике Си. – М., 2003. 4. www.rsasecurity.com.

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua