ПРИМЕНЕНИЕ МЕТОДА ПРЕДСКАЗАНИЯ ВТОРОГО ПОРЯДКА В СИСТЕМАХ УМЕНЬШЕНИЕ ИЗБИТОЧНОСТИ ИНФОРМАЦИИ

Приведены результаты применения метода предсказания второго порядка в системах уменьшение избыточности информации. Проанализировано принцип формирования существенных отсчетов. Осуществлено исследование эффективности предложенного метода в применении к анализу сигналов стандартных форм.

***

УДК 621.391:519.72

Казакова Н.Ф., Щербина Ю.В.

Одесский государственный экономический университет , г. Одесса

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ УНИВЕРСАЛЬНОГО СТАТИСТИЧЕСКОГО ТЕСТА МАУРЕРА ДЛЯ АНАЛИЗА ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Предложена программная реализация универсального статистического теста Маурера для псевдослучайных последовательностей, применяемых в моделировании и криптографии, а также дано его теоретическое обоснование.

Современное развитие вычислительной техники и ее доступность подняли на новый уровень интерес к задачам, решаемым в области моделирования и криптографии. Обе эти области объединяет необходимость использования генераторов случайных чисел, формирующих бинарные последовательности с равномерным законом распределения. При этом требования, предъявляемые к соответствию истинного распределения равномерному, очень высоки.

Постановка проблемы в общем виде. Проблема заключается в том, что всякое вычислительное устройство является устройством, детерминированным изначально, и может формировать только псевдослучайные последовательности, которые могут быть воспроизведены многократно, и вид которых будет определяться заданием начальных параметров программного генератора. Отсюда вытекает задача, которая сводится к построению генератора, формирующего последовательность, внешне не отличающуюся от случайной. Однако для определения степени «похожести» сформированной последовательности на истинно случайную последовательность необходим некоторый объективный критерий.

Выбор подходящего критерия – задача трудоемкая. Впервые подходы к ее решению были описаны Дональдом Кнутом [1]. Он указывал на то, что, хотя в рамках математической статистики создано великое множество критериев равномерности распределения, пользоваться ими следует осторожно. Дело в том, что если испытания, выполненные с применением некоторого критерия, дают положительный ответ на вопрос о случайности последовательности, это еще не означает, что так оно и есть на самом деле. Однако каждая последующая успешная проверка укрепляет уверенность в положительном результате.

Анализ исследований и публикаций. Обычно к последовательности применяется около полудюжины статистических критериев, и если она удовлетворяет этим критериям, то ее считают случайной. В настоящее время для оценки степени «равномерности» распределения вероятностей символов в формируемых псевдослучайных последовательностях (ПСП) обычно используют различные наборы тестов, к числу которых, в первую очередь следует отнести пакет NIST STS [2]. Он включает набор из 16-ти тестов и методику их использования.

Кроме того, известны и другие пакеты тестов, созданные для нужд тестирования. К их числу относится набор статистических тестов под названием Diehard [3], предназначенный для определения качества последовательности случайных чисел. Эти тесты были разработаны Джорджем Марсальей (George Marsaglia). Он включает 12 тестов и доступен по адресу http://stat.fsu.edu/pub/diehard/.

По адресу http://www.isi.qut.edu.au/resources/cryptx/ можно связаться с разработчиками пакета тестов Crypt-X [4] и получить программное обеспечение вместе с руководством по их применению.

Большинство статистических критериев ориентированы на обнаружение в испытуемой ПСП аномалий определенного вида. Этим и объясняется их большое количество, в пакетах, рекомендованных к применению. Кроме того, пакет тестов, как правило, применяют при окончательных испытаниях готового генератора. Однако в процессе проектирования и разработки, необходимо иметь тест, позволяющий оперативно производить оценку генератора, после каждого изменения в алгоритме его работы. Из всего многообразия тестов [2,3,4] наиболее подходящим для этих целей является Универсальный тест Маурера, описанный в [5,6] и входящий в состав пакета [2]. Достоинством этого теста, в отличие от иных тестов, является его способность выявлять широкий спектр аномалий в тестируемом статистическом материале, приводящих к отклонению от равномерного распределения.

Цель статьи. Учитывая сказанное, статья посвящена описанию особенностей программной реализации Универсального теста Маурера.

Изложение основного материала. Тест Маурера основан на идее Зива для универсальных алгоритмов кодирования о том, что случайную последовательность с равномерным законом распределения вероятностей нельзя сколько-нибудь заметно сжать без потери информации. Его тестовая статистика тесно связана с посимвольной энтропией потока, который, по мнению ее автора, является правильной мерой качества формируемой псевдослучайной последовательности.

В соответствии с идеей, лежащей в основе теста, двоичная n-битная последовательность

$$\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n,$$

формируемая на выходе генератора ПСП, разделяется на L-битные непересекающиеся последовательные блоки, размер которых может изменяться от двух до шестнадцати символов. Величина L является параметром, определяемым в процессе тестирования. Он связан с величиной энтропии на символ источника, порождающего ПСП.

После разделения на L-битные блоки тестируемая последовательность делится на два сегмента: сегмент инициализации, включающий Q L-битных блоков и тестируемый сегмент, включающий K следующих за ними блоков такого же размера, так, как это показано на рисунке 1.

Рис. 1 . Разделение тестируемой последовательности

Если длина последовательности n превышает суммарный размер сегментов инициализации и тестирования, то оставшиеся блоки просто отбрасываются и в

тестировании не участвуют. Общая протяженность участка тестируемой последовательности при этом составляет $(Q + K)\ L$ бит.

Далее создается массив размером 2^L, в ячейках которого запоминаются номера последних позиций i, $1 \le i \le Q$, на которых были обнаружены L-разрядные блоки конкретного вида. До начала тестирования все они должны быть заполнены, то есть каждая L-разрядная группа должна встретится на участке инициализации хотя бы один раз. По этой причине величина сегмента инициализации Q выбирается из условия $Q \ge 10 \cdot 2^L$.

Тестирующая функция f_n определяется как средний логарифм (по основанию 2) от интервалов между одноименными L-битными блоками $i - T_j$, где T_j – номер предыдущей позиции, на которой был обнаружен текущий i-й блок. Она имеет следующий вид:

$$f_n = \frac{1}{K} \sum_{i=Q+1}^{Q+K} \log_2 (i - T_j)$$

На основании этой функции, в соответствии с рекомендациями, изложенными в [2], вычисляется показатель

$$P_v = erfc\left(\left| \frac{f_n - m_L}{\sqrt{2}\,\sigma} \right| \right) \qquad (1)$$

где $erfc$ – есть дополнительная функция ошибки, m_L – теоретически ожидаемое выборочное среднее значение, вычисляемое из статистики для данного L-битного блока. Теоретическое значение стандартного отклонения определяется как

$$\sigma = c\sqrt{\frac{D_L}{K}},$$

где

$$c = 0.7 - \frac{0.8}{L} + \left(4 + \frac{32}{L}\right)\frac{K^{-3/L}}{15}.$$

Величины m_L и D_L приведены в [6] и могут быть взяты из следующей таблицы

L	m_L	D_L		L	m_L	D_L
1	0.7326495	0.690		9	8.1764248	3.311
2	1.5374383	1.338		10	9.1723243	3.356
3	2.4016068	1.901		11	10.170032	3.384
4	3.3112247	2.358		12	11.168765	3.401
5	4.2354266	2.705		13	12.168070	3.410
6	5.2177052	2.954		14	13.167693	3.416
7	6.1962507	3.125		15	14.167488	3.419
8	7.1836656	3.238		16	15.167379	3.421

Для выполнения этого теста требуется длинная последовательность двоичных символов, длиной

$$n \ge (Q + K)\ L \qquad (2)$$

Размер блоков L рекомендуется выбирать в пределах $6 < L < 16$.

Размер второго сегмента состоящего из тестируемых блоков, определяется как

$$K = \lceil n/L \rceil - Q \approx 1000 \cdot 2^L.$$

При этом значения величин n, L и Q рекомендуется выбирать следующим образом:

n	L	$Q = 10 \cdot 2^L$		n	L	$Q = 10 \cdot 2^L$
> 387,840	6	640		> 49,643,520	12	40960
> 904,960	7	1280		> 107,560,960	13	81920
> 2,068,480	8	2560		> 231,669,760	14	163840
> 4,654,080	9	5120		> 496,435,200	15	327680
> 10,342,400	10	10240		> 1,059,061,760	16	655360
> 22,753,280	11	20480				

Наиболее предпочтительным размером блока следует, очевидно, считать блок с $L = 8$, поскольку расчет остальных вероятностных показателей формируемых ПСП, таких, например, как энтропия H, производится с привязкой к 256-ти символьной кодовой таблице вычислительного устройства. И, чаще всего, в приложениях именно таким однобайтовым тестом и ограничиваются. Однако, как уже упоминалось, разработчики теста рекомендуют для полноты контроля рассматривать и иные значения L. Это, в свою очередь, вызывает трудности в программной реализации теста. Дело в том, что вычислительные устройства ориентированы на работу с величинами, занимающими в памяти пространства, кратные одному байту, и организовать чтение участков испытуемой последовательности с иными размерами без заметного усложнения алгоритма сложно.

Решение этой проблемы можно найти, реализовав алгоритм тестирования с использованием экономичных с точки зрения затрачиваемых вычислительных ресурсов операций, к числу которых относятся операции сложения и сдвига. В данном случае речь идет о тех операциях, которые задействованы в процедурах обработки потока тестируемой последовательности, поскольку именно они «утяжеляют» вычисления.

В соответствии с рекомендациями NIST [2], величина ошибки первого рода, когда «хорошая» последовательность будет признана «неслучайной», не должна быть больше величины, равной 0.01. Для получения этого показателя необходимо протестировать 100, независимо сформированных испытуемым генератором участков последовательности. Если более чем один из них будет определен как «неслучайный», значит, тест не выполнен и проектируемый генератор требует доработки. При этом под независимостью тестируемых участков ПСП подразумевается, что все они сформированы с различными, случайно выбранными ключами. Общая же длина файла, содержащего испытуемую ПСП, в битах должна составлять величину не менее $100n$, где n определяется выражением (2).

Учитывая, что входной поток должен разделяться на отрезки, которые, в общем случае, не кратны байту, для каждого отдельного значения L потребуется отдельная процедура разделения входного потока. Задача может быть решена следующим образом.

В целое, 64-х битное число записывается некоторое число байт, таким образом, чтобы общее число бит оказалось кратным 8-ми и длине блока L. Например, пусть тест проводится для $L = 7$. Тогда следует последовательно считать из файла 7 байт (56 бит) и записать их в 64 битное число (объявленное как **`unsigned __int64`**), начиная от младшего байта к старшему байту. Далее, последовательно сдвигая полученное число на 7 бит вправо, а затем на 7 бит влево и суммируя полученный результат с его начальным значением, получаем искомый 7-ми битный фрагмент. Повторение этой процедуры 8 раз, позволит считать весь 56-битный отрезок. После этого считывается очередной 7-ми байтовый участок тестируемого фрагмента и так до тех пор, пока этот фрагмент не закончится. Приведенный алгоритм иллюстрируется следующим кодом.

```
            char ch;
            unsigned __int64 a, b;
            unsigned __int8 byt, q, i;
            unsigned __int32  t, tab[128];

               a = 0;
         for (i = 0; i <= 6; i++)
    // Чтение очередных 7-и байтов
               {
                in.get (ch);
                byt = (unsigned __int8) ch;
                b = byt; b = b << (8 * i); a = a ^ b;
               }
         for (i = 1; i <= 8; i ++)
    // Считывание 8-ми 7-битных фрагментов
               {
                b = a; b = b >> 7; b = b << 7; q = a ^ b; a = a >>
7;
                t += 1; tab[q] = t;
    // Запоминание последней позиции, на которой
               }
    // принят данный 7-битный фрагмент
```

Для тестов, использующих другие значения L, процедура выглядит аналогично. При этом для 6-ти битного теста следует выбирать 3-х байтовый сегмент ($6 \times 4 = 24$ бита), для $L = 10$, можно выбрать 5-ти байтовый сегмент ($10 \times 4 = 40$ бит), для $L = 12$, длина сегмента составит 6 байт ($12 \times 4 = 48$ бит) и для $L = 14$, длина сегмента составит 8 байт ($14 \times 4 = 56$ бит).

При $L = 9$ эта задача несколько сложнее. Дело в том, что в пределах 64-х битного числа нет величин, которые были бы кратны одновременно 9-ти и 8-ми. В этом случае предлагается следующее решение. Поскольку ближайшим общим кратным для чисел 8 и 9 является число 72, сначала считывается 8 байт (64 бита) и отдельно считывается и запоминается в виде 64-х битного числа 9-й байт. Затем, после считывания первой 9-ти битовой группы, на освободившееся слева место дописывается 9-й байт. Для этого символы 9-го байта (в 64-х битном представлении) сдвигаются на 55 позиций влево и прибавляются по модулю два к группе двоичных символов, оставшихся от первых 8-ми байт. Далее считываются оставшиеся семь 9-ти битовых групп. Эту процедуру реализует следующий фрагмент кода:

```
            char ch;
            unsigned __int16 q;
            unsigned __int8 byt, i;
            unsigned __int64 a, b, c;
            unsigned __int32 t, tab[512];

               a = 0;
         for (i = 0; i <= 7; i++)
    // Чтение очередных 8-ми байтов
               {
                in.get (ch);
                byt = (unsigned __int8) ch;
                b = byt; b = b << (8 * i); a = a ^ b;
```

```
            }
             in.get (ch);
// Чтение 9-го байта
             byt = (unsigned __int8) ch;
             c = byt; c = c << 55;
          b = a; b = b >> 9; b = b << 9; q = a ^ b; a = a >> 9;
             a = a ^ c;
// Добавление 9-го байта в 55-ю позицию
             t += 1; tab[q] = t;
// Учет одной 9-ти битовой группы
       for (i = 1; i <= 7; i ++)
// Чтение очередных 7-ми 9-ти битовых групп
             {
          b = a; b = b >> 9; b = b << 9; q = a ^ b; a = a >> 9;
             t += 1; tab[q] = t;
             }
```

Для тестов с параметром $L = 11$, $L = 13$ и $L = 15$ задача решается аналогично. Для $L = 8$ и $L = 16$, она вообще не представляет трудности, поскольку их величины кратны одному байту.

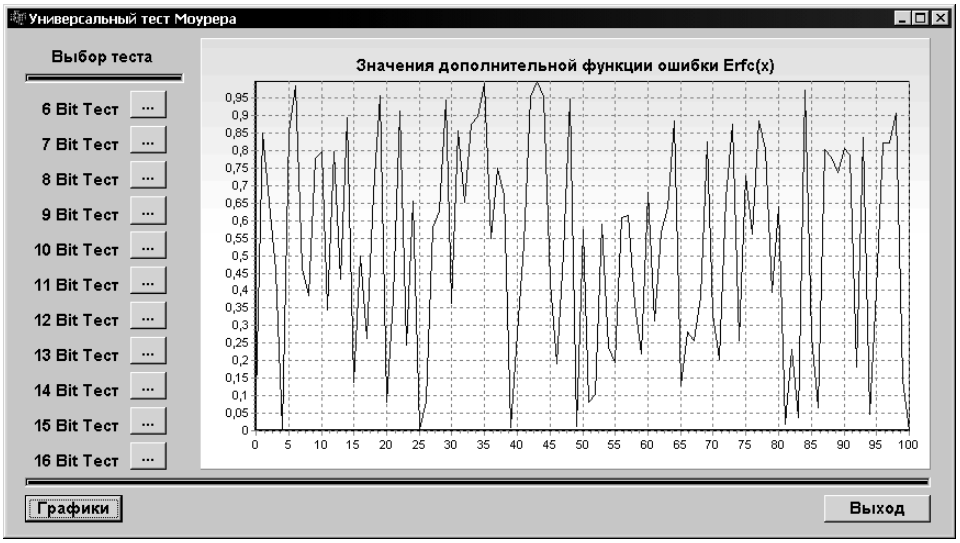

Рис. 2. Внешний вид главной формы

С учетом сказанного авторами разработан программный продукт, реализующий данный тест, который включает одиннадцать отдельных модулей (по числу рекомендованных величин фрагментов L). Он содержит интерфейс, который включает главную форму, позволяющую вызывать требуемый вид теста и отображать графические результаты контроля каждого из 100 тестируемых отрезков (рисунок 2). Для подключения испытуемого файла используются подчиненные формы, вызываемые из главной формы. Реализуемый ими интерфейс позволяет указать путь к файлу и по завершении тестирования отобразить значения функции ошибки $erfc(x)$ и ее аргумент x, определяемых в соответствии с выражением (1) для всех 100 участков тестируемой последовательности.

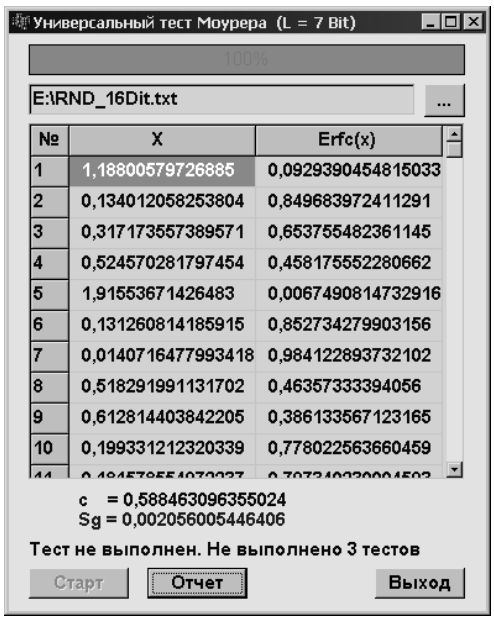

Рис. 3. Внешний вид формы для теста с параметром $L = 7$

По завершении тестирования на форму выводятся значения параметров c и σ. Кроме того, представляются данные об исходе тестирования и количестве участков последовательности, для которых тест не выполнен. На основании величин erfc(x) и x, отображаемых на подчиненной форме строится график, отображаемый на главной форме, что позволяет разработчику получить представление о качестве создаваемого генератора ПСП.

Как уже указывалось, для проведения тестирования создаваемого генератора требуется создать внешний файл, состоящий из ста независимо сформированных отрезков псевдослучайной последовательности. При этом можно для каждого значения L формировать свой отдельный файл, но можно и сформировать один файл, рассчитанный на L = 16. В последнем случае, при переходе к следующему тестируемому фрагменту, надо будет принудительно перемещать указатель позиции на начало очередного фрагмента, если выполняется тест для $L \neq 16$.

Проведенные испытания показали, что тесты для значений $6 < L \leq 10$ выполняются на современном компьютере быстро и без проблем. При больших значениях L затраты вычислительных ресурсов становятся заметными. Достаточно сказать, что при L = 16 на проведение теста затрачивается более часа времени, а длина файла, содержащего тестируемую последовательность, превышает 13 Гб.

Вычисление значения функции ошибки erfc(x) производится путем ее вызова из отдельного программного модуля. Эта процедура может быть реализована по-разному. Один из ее вариантов можно найти на сайте NIST по адресу http://csrc.nist.gov/groups/ST/toolkit /rng/ documentation_software.html.

Выводы

В заключение следует отметить, что испытания различных шифров, проведенные с использованием этого теста, показывают его крайнюю чувствительность к малейшей неравномерности распределения вероятностей символов. Даже незначительные отклонения тестирующей функции fn от ее теоретически ожидаемого выборочное среднее значения mL, ведет к непризнанию тестируемого отрезка последовательности «случайной». С учетом

этого можно сделать вывод об эффективности универсального теста. Его выполнение может служить основанием для роста уверенности в качестве разрабатываемого генератора ПСП.

Перечень использованных источников

1. Кнут Д. Искусство программирования для ЭВМ. Т.2. – М.: Мир, 1977. – 727 с.

2. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. NIST Special Publication 800-22. May 15, 2001.

3. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness // [Электронныйресурс]: http://www.stat.fsu.edu/pub/diehard/

4. Statistical test suite Crypt-X // [Электронныйресурс]: http://www.isi.qut.edu.au/resources/ cryptx/

5. A Universal Statistical Test for Random Bit Generators. Ueli M. Vaurer. Appeared in Jornal of Cryptology, vol. 5, no. 2, 1992, pp. 89-105.

6. Menezes A., van Oorshot P., Vanstone S. Handbook of Applied Cryptography. – CRC Press, 1997

Надійшла в редколегію 21.06.2011

Казакова Н.Ф.,. Щербина Ю.В

ПРОГРАМНА РЕАЛІЗАЦІЯ УНІВЕРСАЛЬНОГО СТАТИСТИЧНОГО ТЕСТУ МАУРЕР ДЛЯ АНАЛІЗУ ПСЕВДОВИПАДКОВОЇ ПОСЛІДОВНОСТІ

Запропонована програмна реалізація універсального статистичного тесту Маурера для псевдовипадкових послідовностей, які використовуються в моделюванні та криптографії. Приводиться його теоретичне обґрунтування.

Ключові слова: тест Маурера; псевдовипадкова послідовність, програма; криптографія.

***

UDK 51-74 : 621.382

W. Wrona[1], V. Karpinskyi[2], M. Bogusz[3]

[1]*Evatronix SA, Poland,,*
[2]*Ternopil Ivan Pul'uj National Technical University and Evatronix SA, Poland,*
[3]*AGH University of Science and Technology, Poland*

ASPECTS OF FAST FOURIER TRANSFORM SIMULATION FOR APPLICATIONS REQUIRING OPERATION IN REAL TIME

Key words: FFT, FPGA, hardware simulation.

This article presents an example of Fast Fourier Transform (FFT) hardware simulation using XILINX evaluation board. The main aspect of this research is FFT implementation in applications which need RTC (Real-Time Clock) digital signal processing. Characteristic properties of the simulation are discussed. This paper considers also an implementation of FFT algorithm in case of signals transformation. The authors present relationship between parameters of Field-Programmable Gate Array (FPGA) unit on evaluation board, parameters of transformation and expected results with included algorithm, especially with time characteristics point of view.

Intoduction

Digital signal transformation and processing are one of the main areas in techniques, which have an important role in science in this century. It is becoming essential not only in state and military sector, but also it reflects market needs [1]. Modern advanced computational methods and algorithms require plenty of operations, and when time is a critical value, the two parameters