ИНФОРМАТИКА

2013 **июль-сентябрь** **№ 3**

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

УДК 004.33.054

С.В. Ярмолик, В.Н. Ярмолик

КВАЗИСЛУЧАЙНОЕ ТЕСТИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Анализируются причинно-следственные связи при возникновении неисправностей вычислительных систем. Даются определения понятий «неисправность», «ошибка» и «неисправное поведение вычислительных систем», показывается их общность для программной и аппаратной частей вычислительных систем. Рассматривается классификация обобщенных входных тестовых воздействий на три категории: точечные тестовые наборы, узкополосные тестовые наборы и блочные тестовые наборы. Приводится анализ методов тестирования вычислительных систем по методике черного ящика, показывается эффективность использования квазислучайного тестирования. Анализируются и предлагаются методы формирования квазислучайных тестовых воздействий.

Введение

В настоящее время актуальной является проблема тестирования современных вычислительных систем, таких как встроенные системы (Embedded System), системы на кристалле (System-on-a-Chip) и сети на кристалле (Net-on-a-Chip) [1, 2]. Характерной особенностью подобных вычислительных систем является тесное взаимодействие их аппаратных и программных средств, причем подавляющую часть аппаратных средств представляют собой запоминающие устройства различного уровня иерархии системы [2, 3]. Согласно последним прогнозным исследованиям менее чем через десять лет встроенная память будет занимать более 95 % площади кристалла вычислительных систем [3].

Архитектурные особенности современных вычислительных систем, многообразие физических дефектов их аппаратной части и ошибок в программном обеспечении, а также многочисленные подходы для тестирования таких систем предопределяют необходимость применения универсальных методов для совместного тестирования аппаратной и программной частей систем [2, 4]. В настоящее время практически отсутствуют методы совместного тестирования аппаратной и программной частей вычислительных систем, в большей части из-за отсутствия точных моделей их неисправного поведения и различной содержательной и терминологической их интерпретации [1, 2, 4, 5].

1. Основные определения и классификация

Большинство терминов и понятий в области тестирования и диагностики вычислительных систем имеют достаточно общий смысл и характеризуются отсутствием точности и однозначности. В основном это связано с разнообразием типов современных вычислительных систем, большим количеством различных методов их тестирования, различными уровнями абстракции их описания, а также с формулировкой различных целей и задач тестирования. Поэтому для последующего обсуждения очень важно однозначно определить следующие термины: физический дефект (physical defect), ошибка проектирования (mistake), неисправность (fault), ошибка системы (error) и неисправное поведение системы (failure).

Обычно на самом низком уровне абстракции используется термин «неисправность» как математическая модель, описывающая физические дефекты системы и ошибки при ее проектировании, изготовлении и использовании. Основными источниками неисправностей являются: ошибки спецификации (specification mistakes) из-за неправильных или неправильно интерпретированных проектных требований к системе и/или ее спецификации; ошибки проектирования (implementation mistakes) в основном из-за ошибок кодирования (bags); физические дефекты аппаратных компонентов системы (component physical defects); внешние факторы (external

factors), такие как условия окружающей среды (температура, электромагнитные и радиационные излучения и др.) или человеческий фактор (ошибки оператора) [1, 4–6]. Тогда неисправность как математическую модель неисправного состояния вычислительной системы определим следующим образом:

Определение 1. Неисправностью вычислительной системы является такая ее функциональность, которая может приводить к новым выходным значениям (наблюдаемому поведению) и/или к новому состоянию системы, не соответствующим спецификации системы и требованиям, предъявляемым к ней.

Согласно определению 1 неисправности вычислительной системы могут приводить к ее неверным наблюдаемым выходным значениям и ошибочным внутренним состояниям или только к ошибочным внутренним состояниям. Отметим, что не для всего множества входных значений наличие неисправности приводит к ошибочным состояниям системы и ошибочным выходным значениям.

Определение 2. Ошибкой вычислительной системы из-за наличия в ней неисправности является формирование для некоторого множества входных значений системы одного или более неверных результатов, отличающихся от ожидаемых значений.

Ошибки вычислительной системы, по сути, являются результатом активизации ее неисправностей. Они могут привести к неисправному поведению системы (системному сбою), которое, в свою очередь, может быть наблюдаемым признаком неисправного поведения (состояния) вычислительной системы.

Определение 3. Неисправным поведением (сбоем) вычислительной системы называется формирование наблюдаемых выходных значений системы, отличных от ожидаемых, для некоторого множества входных значений.

Неисправное поведение вычислительной системы возникает в результате решения задачи транспортировки ошибочного значения (ошибки) системы на ее наблюдаемые выходы.

Приведенные определения в равной степени применимы как к программному, так и аппаратному обеспечению вычислительных систем. Обоснованность и применимость этих определений проиллюстрируем примером из программного обеспечения [6]. Рассмотрим программу, которая вычисляет остаток от деления на три квадрата (*data***2) входного значения *data* (рис. 1).

```
begin
        read (data);
                data: = 2*data;   (← неисправность)
                data: = data mod 3;
        write (data)
end.
```

Рис. 1. Пример программы, содержащей неисправность

В связи с наличием неисправности из-за ошибки при кодировании в третьей строке кода программы эта программа фактически вычисляет остаток от деления на три удвоенного значения *data*. При рассмотрении программы, представленной выше для исходного значения *data* = 3, выходной результат оператора *data*: = 2**data*, содержащего неисправность, составляет 6, тогда как ожидаемое значение, при отсутствии неисправности, должно быть 9. Таким образом, возникает вычислительная ошибка. В то же время для *data* = 2 ошибочного значения не возникает, так как неисправность не активизируется. Кроме того, для обоих значений 3 и 2 входных данных *data* программа вычисляет ожидаемо правильные результаты, а именно 0 и 1. Это означает, что значения *data*, равные 3 и 2, не вызывают неисправного состояния (сбоя программы), так как значение *data* = 2 даже не вызывает ошибку, а значение *data* = 3 инициирует ошибочный промежуточный результат – 6 вместо 9. Однако в обоих случаях формируется правильное выходное значение программы, равное 0. В то же время для *data* = 4 программа, содержащая неисправность в третьем операторе, формирует ошибочное значение (вычислительную ошибку).

Действительно, в этом случае выходное значение равняется 2 вместо ожидаемой величины 1, что свидетельствует о неисправном состоянии программы и приводит к ее сбою.

В следующем примере рассмотрим аппаратную составляющую вычислительной системы, представленную оперативным запоминающим устройством (ОЗУ) [7]. Предположим, что ОЗУ содержит четырехразрядные запоминающие ячейки, которые используются для хранения данных, состоящих из четырех бит и представляющих собой шестнадцатеричную цифру. В одной из ячеек ОЗУ возникла константная неисправность ≡0, которая приводит к тому, что, например, во втором разряде ячейки постоянно находится значение 0 [7]. Что касается остальных разрядов ячейки, то их содержимое может быть 0 или 1. Рассмотрим случай шестнадцатеричных данных, равных 4, 3 и 2, которые должны быть записаны в ОЗУ и затем считаны. Для значения данных 4 наличие неисправности во втором бите ячейки ОЗУ не приводит к ошибке, несмотря на наличие неисправности ≡0, так как $4_{(16)} = 0100_{(2)}$. Оба значения данных 3 и 2 в двоичном представлении (0011, 0010) содержат 1 во втором бите, что является причиной возникновения ошибки выборки после выполнения операции чтения. Это, в свою очередь, может привести к неисправному состоянию вычислительной системы в целом.

Из рис. 2 видно, что первопричиной неисправностей являются: ошибки спецификации, ошибки реализации, физические дефекты аппаратных компонентов системы и внешние факторы [4–6]. Наличие неисправностей может вызывать ошибки как программных, так и аппаратных средств вычислительных систем. Ошибки возникают в результате активизации неисправностей вычислительной системы, как это было показано в приведенных выше примерах. И, наконец, активизированная ошибка вычислительной системы может привести к наблюдаемому неисправному поведению вычислительной системы.

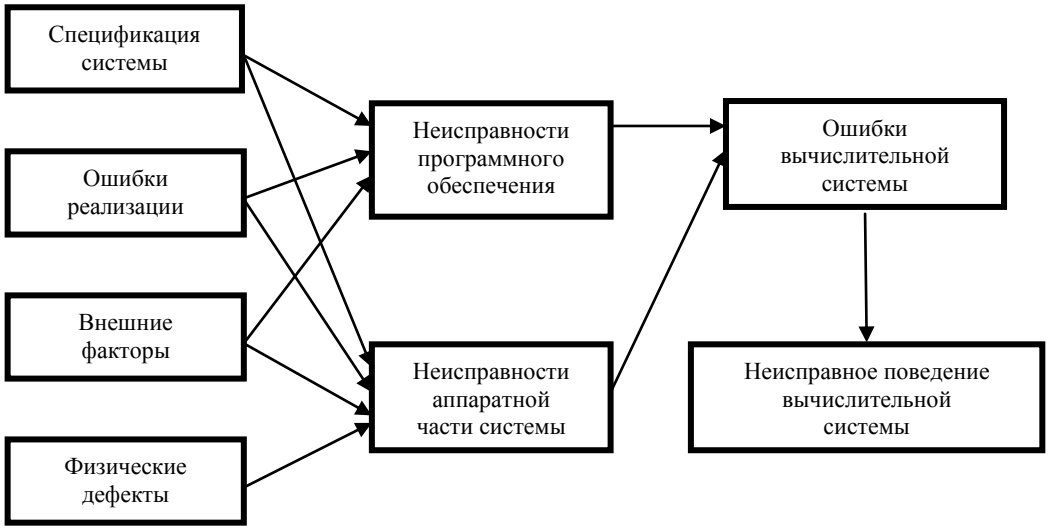


Рис. 2. Причинно-следственная связь между неисправностями, ошибками и неисправным поведением вычислительной системы

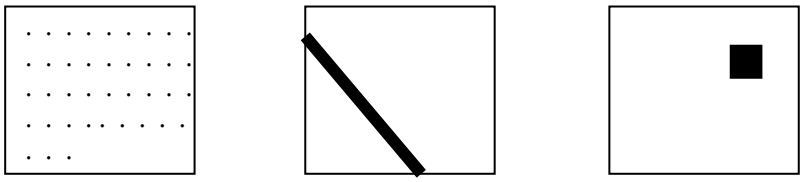
Существующие методологии построения тестовых последовательностей решают задачу нахождения такого подмножества входных тестовых воздействий (failure patterns), которые приводили бы к наблюдаемому неисправному поведению вычислительной системы в случае возникновения в ней неисправностей на всех стадиях жизненного цикла системы [4–6].

2. Обобщенные входные тестовые воздействия

Для того чтобы оценить эффективность различных методов тестирования вычислительных систем, рассмотрим множества их входных тестовых воздействий, которые приводят к наблюдаемому неисправному поведению систем. Очевидно, что данные множества являются уникальными для каждого из объектов тестирования и зависят от многих факторов (см. рис. 2),

а также от его архитектуры и функциональности. Однако структура входных тестовых воздействий характеризуется рядом закономерностей, отмеченных в последних публикациях [8–14].

Большой объем эмпирических и экспериментальных исследований различного рода программного обеспечения показал обобщенность (структурированность) входных воздействий (наборов), которые инициируют неисправное поведение систем, т. е. являются тестовыми воздействиями [8–11]. В основополагающей работе [10] была представлена классификация обобщенных входных тестовых воздействий на три категории: точечные тестовые наборы (point patterns), узкополосные тестовые наборы (strip patterns) и блочные тестовые наборы (block patterns) [10]. Для иллюстрации данной классификации пространство входных наборов рассматривается как двухмерное. Тогда три вида входных тестовых наборов имеют простую геометрическую интерпретацию (рис. 3) [8–11].



а) *б)* *в)*

Рис. 3. Типовые входные тестовые воздействия

На рис. 3 области, выделенные черным цветом, соответствуют входным воздействиям, приводящим к наблюдаемому неисправному поведению программного обеспечения, а фигуры квадратов соответствуют областям входных воздействий [8, 10, 11]. Точечные тестовые наборы (рис. 3, *а*) определяют уникальные входные наборы, распределенные по всему пространству входных наборов, причем их распределение в большинстве случаев имеет регулярную структуру. Узкополосные тестовые наборы (рис. 3, *б*) характеризуются непрерывными множествами входных воздействий имеющих вид узких полос. Тестовые наборы блочного типа (рис. 3, *в*) представляют собой одну либо несколько непрерывных областей входных воздействий, приводящих к наблюдаемому неисправному поведению систем. Как правило, такие типовые входные воздействия выбираются в качестве моделей входных тестовых воздействий, поскольку многочисленные эмпирические исследования подтверждают целесообразность их использования в качестве приближения к реальным входным тестовым воздействиям [8–11]. Несмотря на то что эти модели не являются реальными, их применение, а также применение их комбинаций оказывается весьма эффективным при построении тестовых последовательностей для реальных вычислительных систем [8–11].

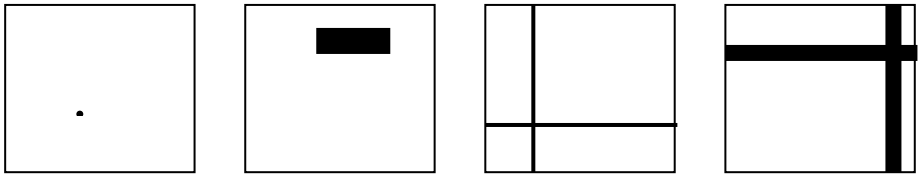
Значительная часть аппаратной составляющей вычислительных систем, особенно систем на кристалле и встроенных систем, включает в себя встроенную память, которая в соответствии с прогнозами будет занимать доминирующую часть на кристаллах вычислительных систем. Поэтому эффективное тестирование памяти и совершенная методология анализа неисправностей запоминающих устройств будет способствовать повышению надежности и выхода годных встроенных систем и систем на кристалле, особенно при ускоренных процессах их разработки и применении передовых технологий изготовления [1, 7].

Причинно-следственный подход может быть использован для прогнозирования неисправного поведения ОЗУ в случае возникновения физических дефектов. Неисправное поведение памяти представляется растровым изображением физического топологического представления результатов тестирования, показывающим расположение неисправных запоминающих ячеек. Чаще всего возникают одиночные неисправности запоминающих ячеек памяти, неисправности групп запоминающих ячеек, неисправности дешифратора адреса ячейки памяти, неисправности шин данных и другие неисправности запоминающих устройств [7, 12–14].

Согласно устоявшейся классификации неисправностей запоминающих устройств различают два их множества: простые, в которых участвуют одна либо две ячейки памяти, и сложные, включающие в себя множество ячеек [7, 14]. К неисправностям, затрагивающим одну ячейку, относятся константные неисправности и переходные неисправности, а к неисправно-

стям, в которых участвуют две ячейки, – неисправности взаимного влияния [7, 14]. Обобщающей моделью входных тестовых воздействий (адресов ячеек памяти) является точечная модель (рис. 4, *а*) либо, для случая многократных неисправностей данных типов, блоковая (рис. 4, *б*) [7, 13, 14]. Кодочувствительные неисправности, неисправности дешифратора адреса и неисправности типа «разрушающая операция чтения» и «ошибочная запись» покрываются моделью узкополосных входных воздействий (рис. 4, *в*) [7, 13, 14]. В случае комбинации физических дефектов памяти, а также их многократных разновидностей входные тестовые воздействия описываются комбинированной моделью (рис. 4, *г*), объединяющей блоковый и узкополосный типы [13].

Таким образом, все множество моделей неисправного поведения запоминающих устройств описывается четырьмя обобщенными моделями входных тестовых воздействий, представленных на рис. 4 [13].



а) *б)* *в)* *г)*

Рис. 4. Типовые входные тестовые воздействия запоминающих устройств

По сравнению с входными тестовыми наборами для программного обеспечения тестовые воздействия запоминающих устройств имеют реальную двухмерную структуру в силу физического двухмерного представления матрицы запоминающих ячеек [14]. В этом случае входными тестовыми данными являются адреса запоминающих ячеек, состоящие из адресов по горизонтальной и вертикальной осям матрицы ячеек памяти.

Как видно из приведенного анализа, типовые входные тестовые воздействия для программного обеспечения и для большей части аппаратных средств вычислительных систем описываются аналогичными моделями, что, очевидно, позволяет использовать единый подход для тестирования вычислительных систем в целом. Следующим важным выводом анализа типовых входных тестовых воздействий является их разнородность по структуре (см. рис. 3 и 4), что затрудняет выбор наиболее эффективного метода их тестирования.

3. Анализ методов тестирования вычислительных систем

В общем случае тестирование вычислительных систем направлено на повышение их надежности и заключается в выявлении максимально возможного количества неисправностей систем за приемлемый (реальный) промежуток времени. Исчерпывающее тестирование (exhaustive testing) характеризуется максимально возможной эффективностью, так как при его реализации проверяется объект тестирования на всевозможных входных воздействиях для всех возможных состояний объекта тестирования [15]. В случае двоичных входных воздействий из N бит необходимо сгенерировать 2^N двоичных комбинаций. Всевозможные двоичные комбинации генерируются для каждого состояния, которое в случае цифровых устройств и систем определяется количеством M запоминающих элементов, которые используются для построения регистров, счетчиков и других последовательностных устройств систем. Таким образом, количество состояний цифрового устройства или системы определяется величиной 2^M в силу того, что каждый запоминающий элемент может находиться в одном из двух состояний, а именно в состоянии 0 или состоянии 1. Сложность исчерпывающего теста (количество тестовых наборов) определяется астрономической величиной 2^{N+M}. Экспоненциальный рост длины исчерпывающего теста существенно ограничивает область его применения только для случаев простейших вычислительных устройств с небольшими значениями N и M [15, 16]. Локально исчерпывающее (locally exhaustive) [16] или псевдоисчерпывающее (pseudo exhaustive) [17, 18] тестирование вычислительных систем позволяет избежать ограничений на число входов тестируемого устройства или системы. Эти подходы являются реальной альтернативой для исчерпы-

вающего тестирования и позволяют существенно сократить число тестовых векторов, однако требуют большого объема исследований при построении подобных тестов [16, 18], а также детального описания вычислительных систем, что не всегда возможно.

Для эффективной аппроксимации исчерпывающего и псевдоисчерпывающего тестирования широко используется случайное (вероятностное) тестирование (random testing) [19]. В данном случае под аппроксимацией понимается формирование заведомо не исчерпывающих и не псевдоисчерпывающих тестов, а тестов, которые являются некоторым их приближением. Данный метод тестирования широко применяется в рамках модели черного ящика (black box), когда вычислительная система описывается только на уровне выполняемых ею функций, а ее внутренняя структура и конкретная реализация как программной, так и аппаратной частей не учитываются. Согласно определению случайного тестирования очередное значение тестового набора выбирается случайным образом, независимо от значений предыдущих наборов теста [19, 20]. Случайное тестирование является неэффективным, когда плотность оставшихся (необнаруживаемых) неисправностей оказывается низкой [20]. Случайное тестирование не использует информацию, которая доступна при реализации метода черного ящика, а именно информацию о предыдущих тестовых наборах, что увеличивает длину тестовой последовательности и уменьшает полноту покрытия неисправностей вычислительных систем [11, 18, 21]. Использование информации о предыдущих тестовых наборах явилось основой создания так называемого антислучайного тестирования (antirandom testing) [18, 22, 23].

Предпосылкой антислучайного тестирования является то, что для достижения более высокой полноты покрытия неисправностей вычислительных систем очередной тестовый набор выбирается из случайных тестовых наборов таким образом, чтобы он был максимально отличным от ранее сгенерированных наборов. Для этого используются различные метрики отличия, такие как расстояние Хемминга и декартово расстояние [22, 23]. При антислучайном тестировании очередной тестовый набор выбирается из множества наборов с максимальным расстоянием от ранее сгенерированных наборов, что позволяет достичь большей эффективности по сравнению со случайным тестированием [22, 23]. К сожалению, основным недостатком антислучайного тестирования является его вычислительная сложность, вызванная необходимостью вычисления метрик расстояния для каждого кандидата в тесты [22]. Даже для улучшенных вариантов антислучайного тестирования вычислительная сложность их реализации является существенным ограничением при практическом применении для реальных размерностей тестовых наборов [23]. В качестве более эффективной метрики для построения антислучайного теста с ограниченным числом наборов используется максимально минимальное расстояние Хэмминга (maximal minimal hamming distance), которое применяется для построения антислучайного теста [18, 25, 26]. К сожалению, проблема вычисления расстояния является ключевой проблемой при генерировании антислучайных тестов, что ограничивает области их применения [11, 18, 22–26].

Все перечисленные методы тестирования в своей основе используют модель черного ящика и направлены на формирование входных тестовых наборов как подмножества входных воздействий, формируемых в большинстве случаев с использованием случайных воздействий [15–26]. Случайное тестирование и все многообразие его модификаций можно описать в терминах численных методов Монте-Карло, основанных на получении большого числа реализаций стохастического (случайного) процесса. Основными недостатками методов случайного тестирования является их невысокая полнота покрытия неисправностей и большая длина тестовых последовательностей. Подобными недостатками характеризуется и метод Монте-Карло, для которого характерны значительная вычислительная погрешность и заметная временная сложность. Поэтому в качестве альтернативного решения для тестирования вычислительных систем в настоящей статье предлагается использовать идеи квазислучайного тестирования, основанного на применении квазислучайных последовательностей как тестовых последовательностей, эффективно покрывающих пространство входных воздействий систем [27–31]. Аналогично, как и в методе квази-Монте-Карло (КМК), квазислучайное тестирование, очевидно, позволит достичь большей полноты покрытия при меньших длинах тестовых последовательностей. Основой реализации квазислучайного тестирования являются квазислучайные последовательности. В следующем разделе статьи проводится анализ подобных последовательностей, обосновыва-

ется использование последовательностей Соболя и предлагаются авторские модификации для генерирования большого числа их реализаций.

4. Квазислучайные последовательности

Последовательность неслучайных чисел называется псевдослучайной последовательностью чисел, если она обладает всеми свойствами случайной последовательности [32]. Последовательность неслучайных чисел называется квазислучайной, если ее можно использовать в реализации алгоритмов Монте-Карло вместо случайной последовательности [33]. Именно такие последовательности используются на практике для различных задач метода КМК, что позволяет достичь меньших вычислительных погрешностей и более быстрой сходимости [27, 33, 34]. Это достигается не столько свойством независимости, характерным для псевдослучайных последовательностей, сколько свойством равномерности. В русскоязычной литературе такие последовательности называют согласно Соболю [27, 33, 34] ЛП$_\tau$-последовательностями, что интерпретируется как «любой последовательный участок хорошо распределен» (более равномерно по сравнению с псевдослучайными последовательностями), в англоязычной литературе – последовательностями с малым дискрепансом (low-discrepancy sequence), а их разновидности – по именам авторов, выделяя последовательности Соболя [27–31].

Для визуальной демонстрации большей равномерности квазислучайной последовательности точек по сравнению с псевдослучайной последовательностью обычно рассматривается двухмерное пространство в виде единичного квадрата. Это пространство равномерно делится на подквадраты. Так, например, квадрат на рис. 5 равномерно разбит на 64 подквадрата и на них нанесены 64 квазислучайные точки ЛП$_\tau$-последовательности (рис. 5, *а*) и 64 псевдослучайные точки (рис. 5, *б*). Из приведенных рисунков видно, что в каждый подквадрат попало ровно по одной квазислучайной точке, в то время как для псевдослучайных точек равномерное заполнение подквадратов не выполняется.

а) *б)*

Рис. 5. Множества точек в двухмерном пространстве: *а)* квазислучайных; *б)* псевдослучайных

Следует отметить, что данное свойство ЛП$_\tau$-последовательностей выполняется для пространств произвольной размерности, а не только для двухмерного пространства [33, 34].

4.1. Последовательности Корпута

Последовательности Корпута являются простейшим примером квазислучайных последовательностей [27, 28]. Для произвольного целого $n \in \{1, 2, 3, \ldots, N\}$ значение элемента x_n последовательности Корпута может быть получено для любого базиса p представления числа n, где p – простое целое число. Первоначально целое число n представляется в базисе p как

$$n = \sum_{i=0}^{I} \alpha_i(n)\, p^i,$$

где $\alpha_i(n) \in \{0, 1, 2, \ldots, p-1\}$ является значением i-й цифры p-ичного представления числа n, а I – наименьшим целым значением i, для которого $\alpha_i(n) \neq 0$, а для всех $i > I$ выполняется равенство $\alpha_i(n) = 0$. Значение I вычисляется как

$$I = \lfloor \log_p n \rfloor, \quad n = \overline{1, N}.$$

Затем для получения значения x_n цифры $\alpha_i(n)$ числа n транспонируются относительно запятой, разделяющей целую и дробную части p-ичного числа, таким образом, что первой после запятой оказывается младшая $\alpha_0(n)$ цифра p-ичного представления числа n. Следующей цифрой будет $\alpha_1(n)$ и т. д. Аналитически процедура транспонирования, в результате которой получается значение x_n, описывается формулой

$$x_n = \sum_{i=0}^{I} \frac{\alpha_i(n)}{p^{i+1}}.$$

В качестве примера рассмотрим целое число $n = 11$, которое представим в базисе $p = 3$ как $11_{(10)} = 1\times3^2 + 0\times3^1 + 2\times3^0 = 102_{(3)}$ с $\alpha_0(11) = 2$, $\alpha_1(11) = 0$ и $\alpha_2(11) = 1$. Следует отметить, что $I = \lfloor \log_3 11 \rfloor = 2$ и соответственно $\alpha_i(n)=0$ для $i>2$. Выполнив процедуру транспонирования, получим значение элемента x_{11} последовательности Корпута $x_{11} = 2/3^1 + 0/3^2 + 1/3^3 = 19/27$, которое принадлежит интервалу [0, 1]. Квазислучайной последовательностью Корпута с основанием $p = 2$ для $N = 7$ является последовательность x_n, состоящая из следующих элементов: $x_1 = 1/2_{(10)}=0{,}1_{(2)}$; $x_2 = 1/4_{(10)}=0{,}01_{(2)}$; $x_3 = 3/4_{(10)}=0{,}11_{(2)}$; $x_4 = 1/8_{(10)}=0{,}001_{(2)}$; $x_5 = 5/8_{(10)}=0{,}101_{(2)}$; $x_6 = 3/8_{(10)}=0{,}011_{(2)}$ и $x_4 = 7/8_{(10)}=0{,}111_{(2)}$.

Важным свойством последовательности Корпута является то, что после генерирования каждого множества из 2^k–1 элементов для приведенного примера (k = 1, 2 и 3) последовательность максимально равномерно распределена, т. е. самый длинный интервал, который не содержит элементов из последовательности Корпута, является минимальным.

Использование произвольного базиса p предопределяет формирование иррациональных дробных чисел, что затрудняет применение подобных последовательностей для целей тестирования вычислительных систем при $p \neq 2$.

4.2. Последовательности Халтона

Последовательности Халтона представляются последовательностью точек, задаваемых их координатами, в s-мерном ($s \geq 1$) пространстве [29]. Они являются наиболее известными многомерными квазислучайными последовательностями. Данные последовательности служат основой для построения других разновидностей квазислучайных последовательностей [29]. Первая координата точек Халтона задается последовательностью Корпута с основанием $p = 2$, вторая координата определяется последовательностью Корпута с основанием 3. В общем случае координаты точек Халтона задаются последовательностями Корпута с использованием простых чисел p как оснований систем счисления [29].

В качестве примера рассмотрим последовательность точек Халтона в двухмерном пространстве (s=2) для $p = 2$ и 3. Для построения данной последовательности необходимо использовать две последовательности Корпута для p = 2 и 3. В десятичной системе счисления эти последовательности иррациональных чисел принимают следующий вид: 1/2, 1/4, 3/4, 1/8, 5/8, 3/8, 7/8, ... и 1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, ... Тогда последовательность Халтона будет представляться последовательностью точек (1/2, 1/3), (1/4, 2/3), (3/4, 1/9), (1/8, 4/9), (5/8, 7/9), (3/8, 2/9), (7/8, 5/9), ... двухмерного пространства. Пример последовательности точек Халтона x_n для трехмерного случая приведен в табл. 1.

Таблица 1

Последовательность Халтона

x_n	p=2	p=3	p=5
n=1	1/2	1/3	1/5
n=2	1/4	2/3	2/5
n=3	3/4	1/9	3/5
n=4	1/8	4/9	4/5
n=5	5/8	7/9	1/25
n=6	3/8	2/9	6/25
n=7	7/8	5/9	11/25
n=8	1/16	8/9	16/25

Отмечается, что последовательности Халтона характеризуются низким качеством для размерностей *s*, больших чем 14 [29]. На практике в силу корреляционных зависимостей последовательности Халтона используются для значений *s*, не больших чем 6 [29].

Так же, как и в случае последовательностей Корпута, применение последовательностей Халтона для целей технической диагностики практически ограничивается только одномерными последовательностями, когда $s = 1$ и соответственно $p = 2$.

4.3. Последовательности Соболя

Последовательности Соболя используют двоичную систему счисления для формирования координат точек в *s*-мерном пространстве и в силу данного обстоятельства являются широко востребованными для современных приложений [27, 29–31]. Последовательность точек Соболя в *s*-мерном пространстве строится на основании одномерной последовательности Корпута, когда первая координата точек формируется последовательностью Корпута для $p = 2$, а остальные – путем процедуры перестановок [27]. При этом перестановки зависят от так называемых *направляющих чисел* (*direction numbers*) v_i^j, применяемых для всех измерений $j = \overline{1, s}$ *s*-мерного пространства. Направляющие числа $\mathrm{v}_i^j = \frac{m_i^j}{2^i}, i = \overline{1, w}$, образуют последовательность дробных двоичных чисел с *w* двоичными битами после запятой, где m_i^j представляет собой целое нечетное число, удовлетворяющее неравенству $0 < m_i^j < 2^i$. Для описания конкретной последовательности Соболя необходимо задать направляющие числа v_i^j для всех измерений *s*-мерного пространства.

Значение *n*-го элемента (точки) x_n последовательности Соболя определяется его координатами x_n^j для всех измерений *s*, которые вычисляются согласно выражению

$$x_n^j = \alpha_0(n)\mathrm{v}_1^j \oplus \alpha_1(n)\mathrm{v}_2^j \oplus \ldots \oplus \alpha_{w-1}(n)\mathrm{v}_w^j, \quad j = \overline{1, s}.$$

Здесь $\alpha_i(n) \in \{0, 1\}$, $i = \overline{0, w-1}$, являются значениями цифр двоичного представления $\sum_{i=0}^{\lfloor \log_2 n \rfloor} \alpha_i(n) 2^i$ числа *n*, а символ $\oplus$ означает операцию поразрядного сложения по модулю два. Например, если $m_1^j = 1$, $m_2^j = 3$, $m_3^j = 7$ (w=3) и соответственно v_1^j =0,100, v_2^j= 0,110 и v_3^j= 0,111, можно получить значение произвольного элемента последовательности Соболя для $n \in \{1, 2, \ldots, 2^w - 1\} = \{1, 2, \ldots, 7\}$. Предположим, что $n = 7_{(10)}$, которое в двоичном представлении записывается в виде $111_{(2)}$, тогда значение *j*-й координаты x_7^j седьмого элемента x_7 последовательности Соболя вычисляется как

$$x_7^j = \alpha_0(7)\mathrm{v}_1^j \oplus \alpha_1(7)\mathrm{v}_2^j \oplus \alpha_2(7)\mathrm{v}_3^j = 0{,}100 \oplus 0{,}110 \oplus 0{,}111 = 0{,}101.$$

Все элементы последовательности Соболя длиной $2^w - 1 = 2^3 - 1 = 7$ приведены в табл. 2.

Таблица 2

Одномерная классическая последовательность Соболя

$n_{(10)}$	$n_{(2)} = \alpha_2(n)\alpha_1(n)\alpha_0(n)$	x_n	$2^w \times x_n$
1	0 0 1	v_1	1 0 0
2	0 1 0	v_2	1 1 0
3	0 1 1	$\mathrm{v}_1 \oplus \mathrm{v}_2$	0 1 0
4	1 0 0	v_3	1 1 1
5	1 0 1	$\mathrm{v}_1 \oplus \mathrm{v}_3$	0 1 1
6	1 1 0	$\mathrm{v}_2 \oplus \mathrm{v}_3$	0 0 1
7	1 1 1	$\mathrm{v}_1 \oplus \mathrm{v}_2 \oplus \mathrm{v}_3$	1 0 1

Здесь для одномерного случая $v_i^1 = v_i$, а $x_n^1 = x_n$.

Последовательности Соболя, основанные на использовании двоичной системы счисления ($p = 2$), являются широко востребованной разновидностью квазислучайных последовательностей чисел в основном из-за удобства реализации на ЭВМ алгоритмов их генерирования. Основными недостатками классической последовательности Соболя являются заметная вычислительная сложность, зависящая от значения номера элемента (количества операций сложения по модулю два), а также ограниченное множество последовательностей, определяемое выбором направляющих чисел [29–31, 33, 34].

5. Модифицированные последовательности Соболя

Из табл. 2 видно, что значение координаты n-го элемента x_n последовательности Соболя вычисляется как поразрядная сумма по модулю два до $w=\lfloor \log_2 n \rfloor$ операндов в зависимости от количества ненулевых компонентов двоичного представления $\alpha_{w-1}(n)\alpha_{w-2}(n)\ldots\alpha_1(n)\alpha_0(n)$ величины n. Количество операндов может быть снижено до одного при использовании кода Грея [35]. Известно, что двоичное число $n+1$, закодированное в коде Грея, отличается от числа n только в одном бите. Представление числа n в коде Грея может быть получено согласно известному соотношению $n_g=g_{w-1}(n)g_{w-2}(n)\ldots g_1(n)g_0(n)=\alpha_{w-1}(n)\alpha_{w-2}(n)\ldots\alpha_1(n)\alpha_0(n)\oplus 0\alpha_{w-1}(n)\alpha_{w-2}(n)\ \ldots\ \alpha_1(n)$ [35]. Здесь индекс g числа n_g означает его представление в коде Грея.

В силу того что $(n+1)_g$ отличается от n_g только в одном бите, значение $x_{(n+1)g}^j$ будет отличаться от величины x_{ng}^j только значением одного направляющего числа v_i^j. Тогда значение j-й координаты $x_{(n+1)g}^j$ элемента $x_{(n+1)g}$ последовательности Соболя будет определяться как

$$x_{(n+1)g}^j = x_{ng}^j \oplus v_i^j . \qquad (1)$$

Соотношение (1) является описанием экономичного способа Антонова – Салеева для формирования последовательностей Соболя, приводимого во многих источниках, в том числе и в [28]. Процедура формирования последовательности Соболя для одномерного случая в соответствии с (1) представлена в табл. 3. Здесь рассмотрен случай последовательности, сгенерированной при условиях, которые аналогичны последовательности, приведенной в табл. 2.

Таблица 3

Одномерная последовательность Соболя, сгенерированная по способу Антонова – Салеева

n	n_g	v_i	$2^w \times x_{ng}$
0 0 1	001⊕000=001	v_1	100
0 1 0	010⊕001=011	v_2	010
0 1 1	011⊕001=010	v_1	110
1 0 0	100⊕010=110	v_3	001
1 0 1	101⊕010=111	v_1	101
1 1 0	110⊕011=101	v_2	011
1 1 1	111⊕011=100	v_1	111

Значение индекса направляющего числа v_i^j в выражении (1) зависит от так называемой последовательности переключений T_q отраженного кода Грея [35]. Для классического отраженного кода Грея переключательная последовательность задается рекурсивной процедурой следующим образом. Первоначально задается $T_1 = 1$ и, если $q > 1$, $T_q = T_{q-1}, q, T_{q-1}$. Например, для $q = 4$ получим $T_4 = 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1$.

Применение соотношения (1) позволяет существенно снизить вычислительную сложность генерирования последовательностей Соболя, что и предопределило их широкое применение на практике [28, 30, 31].

Вторым недостатком последовательностей Соболя является ограниченность их количества, которое определяется наборами направляющих чисел. С целью расширения множества последовательностей Соболя введем формализацию представления модифицированных направляющих чисел в виде нижней треугольной матрицы (унитреугольной матрицы) с единичной главной диагональю.

Первоначально отметим, что в силу ранее приведенных ограничений направляющие числа для всех измерений $v_i=0,\beta_{-1}(i)\beta_{-2}(i)\ldots\beta_{-w}(i)$ во всех случаях имеют определенные значения $\beta_{-i}(i)=1$ и $\beta_{-j}(i)=0$ для $j>i$, так же, как и произвольные значения $\beta_{-j}(i)\in\{0,1\}$ для $j<i$. Это означает, что для всех возможных последовательностей Соболя и всех их координат $v_1 = 0{,}100\ldots0$, а соответственно $2^w\times v_1=100\ldots00$ в силу того, что для m_1 существует только одно безальтернативное значение $m_1=1<2^1$. Так как m_2 есть нечетное целое, меньшее чем 2^2, оно может принимать два значения: 1 или 3. Соответственно $2^w\times v_2=\beta_{-1}(2)10\ldots00$, где $\beta_{-1}(2)=0$ для $m_2=1$ и $\beta_{-1}(2)=1$ для $m_2=3$. Для m_3 имеем $2^w\times v_3=\beta_{-1}(3)\beta_{-2}(3)10\ldots00$ и т. д.

Далее рассмотрим случай одномерных последовательностей Соболя и обозначим значения $2^w\times v_i$ новой переменной μ_i, которую будем рассматривать как значения модифицированных направляющих чисел. Отметим, что результаты, полученные для одномерного случая, легко обобщаются на многомерные последовательности Соболя.

Числа μ_i можно представить в виде нижней треугольной матрицы (унитреугольной матрицы) с единичной главной диагональю (табл. 4).

Таблица 4

Значения модифицированных направляющих чисел

μ_i	$\beta_{-1}(i)$	$\beta_{-2}(i)$	$\beta_{-3}(i)$	…	$\beta_{-w+1}(i)$	$\beta_{-w}(i)$
μ_1	1	0	0	…	0	0
μ_2	$\beta_{-1}(2)$	1	0	…	0	0
μ_3	$\beta_{-1}(3)$	$\beta_{-2}(3)$	1	…	0	0
…	…	…	…	…	…	…
μ_{w-1}	$\beta_{-1}(w-1)$	$\beta_{-2}(w-1)$	$\beta_{-3}(w-1)$	…	1	0
μ_w	$\beta_{-1}(w)$	$\beta_{-2}(w)$	$\beta_{-3}(w)$	…	$\beta_{-w+1}(w)$	1

Согласно процедуре генерирования $x_{(n+1)g} = x_{ng}\oplus\mu_i$ последовательности Соболя (1) для получения очередного значения используется только одно направляющее число. Индекс i направляющего числа μ_i определяется последовательностью переключений отраженного кода Грея, в которой на каждой второй позиции используется индекс 1, на каждой четвертой – индекс 2, на каждой восьмой – 3 и т. д. Это следует из определения переключательной последовательности и видно из ранее приведенного примера для T_4 [35]. Таким образом, каждое второе значение последовательности Соболя получается с использованием μ_1, каждое четвертое с использованием μ_2 и т. д. Для μ_1 значение $\beta_{-1}(1)=1$, что обеспечивает максимальную частоту изменения старшего бита кода элемента x_{ng} последовательности Соболя. Максимальная частота изменения следующего бита кода x_{ng} в два раза меньше и т. д. (см. табл. 3).

Следует отметить, что наиболее важным элементом последовательности Соболя являются направляющие числа μ_i, $i=\overline{1,w}$, приведенные в табл. 4, которые должны принимать уникальные значения для каждой координаты точек в s-мерном пространстве. Наиболее значимым свойством направляющих чисел, вытекающим из ограничений на эти числа и подтверждающимся видом введенной авторами матрицы модифицированных порождающих чисел, является их линейная независимость.

Для получения полного множества модифицированных направляющих чисел μ_i, $i=\overline{1,w}$, на основании $m<w$ исходных, так же, как и в оригинальном методе Соболя, будем использовать примитивные порождающие полиномы. В общем виде примитивный полином имеет вид $\varphi(x)=1\oplus\lambda_1x^1\oplus\lambda_2x^2\oplus\ \ldots\ \oplus\lambda_{m-1}x^{m-1}\oplus x^m$, где $m=\deg\varphi(x)$, а двоичные коэффициенты $\lambda_i\in\{0,1\}$, $i=\overline{1,m-1}$, определяют конкретный вид полинома [32]. Подобный подход, когда на основании

m исходных направляющих чисел и порождающего полинома $\varphi(x)$ генерируются остальные $w–m$ числа, представлен в оригинальном методе Соболя и его классических модификациях, в том числе и в экономичном способе Антонова – Салеева.

Рассмотрим алгоритм формирования модифицированных направляющих чисел, представленных в виде нижней треугольной матрицы с единичной главной диагональю (см. табл. 4).

При реализации каждой итерации значение предыдущих (ранее полученных) направляющих чисел будем модифицировать с целью формирования двоичных кодов, разрядность которых увеличивается от первоначальных m бит до результирующих w бит. Формально это достигается сдвигом направляющего числа на один разряд влево с одновременным приписыванием в младшем разряде двоичного нуля. Подобная модификация математически записывается как

$$\mu_j = \mu_j \times 2^1, \quad j = \overline{1,i}, \quad m \le i < w.$$

Значение μ_j в левой части равенства есть новое значение направляющего числа μ_j, полученное на основании его предыдущего значения. Само рекуррентное соотношение примет вид

$$\mu_{i+1} = \lambda_1\mu_i \oplus \lambda_2\mu_{i-1} \oplus \ldots \oplus \lambda_{m-1}\mu_{i-m+2} \oplus \mu_{i-m+1} \oplus (\mu_{i-m+1} / 2^m). \qquad (2)$$

Значение $\mu_{i-m+1}/2^m$ представляет собой сдвинутую копию вправо на m позиций двоичного кода μ_{i-m+1}.

Например, для случая последовательности Соболя длиной $2^w–1=2^5–1=31$, где $w=5$, используем примитивный полином $\varphi(x)=1\oplus x^1\oplus x^3$ степени $m=3$, а для первых $m=3$ направляющих чисел возьмем целые нечетные числа $m_1 = 1$, $m_2 = 3$ и $m_3 = 5$. Тогда v_1 =0,100, v_2 = 0,110, v_3 = 0,101 и соответственно $\mu_1 = v_1 2^3 = 100$, $\mu_2 = v_2 2^3 = 110$ и $\mu_3 = v_3 2^3 = 101$. Так как $w=5$, остальные $(w–m)$ направляющие числа, в данном случае 5–3=2 числа, генерируются с использованием рекуррентного соотношения (2) для заданных значений m и w:

$$\mu_j = \mu_j \times 2^1, \quad j = \overline{1,i}, \quad 3 \le i < 5;$$

$$\mu_{i+1} = \mu_i \oplus \mu_{i-2} \oplus (\mu_{i-2} / 2^3).$$

Для $i=3$ получим $\mu_1=\mu_1\times2=1000$, $\mu_2=\mu_2\times2=1100$ и $\mu_3=\mu_3\times2=1010$, кроме того, $\mu_1/2^3=0001$. Тогда $\mu_4=\mu_3\oplus\mu_1\oplus(\mu_1/2^3)=1010\oplus1000\oplus0001=0011$, и далее для $i=4$ получим $\mu_5=\mu_4\oplus\mu_2\oplus(\mu_2/2^3)$ = = 00110 ⊕ 11000 ⊕ 00011 = 11101.

Последовательность Соболя, соответствующая полученным направляющим числам μ_1=10000, μ_2=11000, μ_3=10100, μ_4=00110 и μ_5=11101, приведена в табл. 5.

Таблица 5

Последовательность Соболя

n	T_q	x_{ng}	n	T_q	x_{ng}	n	T_q	x_{ng}	n	T_q	x_{ng}
1	1	10000	9	1	00010	17	1	01011	25	1	11001
2	2	01000	10	2	11010	18	2	10011	26	2	00001
3	1	11000	11	1	01010	19	1	00011	27	1	10001
4	3	01100	12	3	11110	20	3	10111	28	3	00101
5	1	11100	13	1	01110	21	1	00111	29	1	10101
6	2	00100	14	2	10110	22	2	11111	30	2	01101
7	1	10100	15	1	00110	23	1	01111	31	1	11101
8	4	10010	16	5	11011	24	4	01001			

Второй модификацией последовательностей Соболя является использование в качестве направляющих чисел w последовательных значений М-последовательности, формируемых в соответствии с примитивным порождающим полиномом степени w. В отличие от классического метода формирования направляющих чисел в данном случае генерируется все множество w чисел, а не только $w–m$, на основании m исходных чисел.

Для соблюдения всех свойств последовательностей Соболя необходимо, чтобы μ_1 всегда равнялось w-разрядному коду 100…0 (см. табл. 4). Для получения всего множества модифицированных направляющих чисел двоичный код 100…0 используем в качестве начального состояния генератора M-последовательности и сформируем w–1 последующих состояний генератора, которые будем использовать как $\mu_2, \mu_3, \ldots, \mu_w$.

Например, для полинома $\varphi(x) = 1 \oplus x^1 \oplus x^4$ степени $m = 4$ направляющие числа принимают значения $\mu_1 = 1000$, $\mu_2 = 1100$, $\mu_3 = 1110$ и $\mu_4 = 1111$. Изменив порождающий полином на $\varphi(x) = 1 \oplus x^3 \oplus x^4$, получим новое множество модифицированных направляющих чисел, а именно $\mu_1 = 1000$, $\mu_2 = 0100$, $\mu_3 = 0010$ и $\mu_4 = 1001$. Для двух приведенных примеров соответствующие треугольные матрицы V_1 и V_2 с единичной главной диагональю, построенные на основании модифицированных направляющих чисел, имеют вид

$$V_1 = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{vmatrix}; \qquad V_2 = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{vmatrix}.$$

Для каждого множества направляющих чисел, описываемых матрицами V_1 и V_2, может быть сформирована уникальная последовательность Соболя.

Очевидно, что количество множеств модифицированных порождающих чисел определяется количеством примитивных порождающих полиномов степени w. Для общего случая это количество вычисляется как $\Phi(2^w–1)/w$, где Ф есть функция Эйлера от целого числа $2^w–1$ [32].

Следующей модификацией последовательностей Соболя является использование для формирования произвольных значений $\beta_{–j}(i) \in \{0,1\}$ для $j < i$ последовательных $(w^2–w)/2$ бит произвольной псевдослучайной последовательности, в том числе и M-последовательности. Возможность формирования подобным образом модифицированных направляющих чисел μ_i следует из того факта, что равенство единице младшего бита двоичного представления целого числа свидетельствует о его нечетности. Это отражено, например, в табл. 4 в виде равенства $\beta_{–i}(i) = 1$, $i = \overline{1,w}$. Очевидно, что старшие биты нечетного целого числа могут принимать произвольные значения.

В случае модифицированных направляющих чисел, как отмечалось ранее, значение μ_1 всегда равняется w-разрядному коду 100…00. Соответственно $\mu_2 = \beta_{–1}(2)10\ldots00$, где $\beta_{–1}(2) \in \{0, 1\}$, $\mu_3 = \beta_{–1}(3)\beta_{–2}(3)10\ldots00$, где $\beta_{–1}(3)$ и $\beta_{–2}(3) \in \{0, 1\}$ и т. д. Значение $\mu_w = \beta_{–1}(w)\beta_{–2}(w)\beta_{–3}(w) \ldots \beta_{-w+1}(w)1$, где $\beta_{–j}(w) \in \{0,1\}$ для $j < w$. Таким образом, $\beta_{–j}(i) \in \{0,1\}$ при $j<i$ для модифицированных направляющих чисел μ_i могут принимать произвольные двоичные значения. Количество возможных вариантов значений $\beta_{–j}(i) \in \{0,1\}$ при $j < i$ определяется их числом $(w^2–w)/2$ и вычисляется как $2^{(w^2-w)/2}$. Это подтверждается анализом табл. 4 и матриц V_1 и V_2.

В качестве примера рассмотрим случай, когда w=4. Тогда число значений $\beta_{–j}(i) \in \{0,1\}$ при $j < i$, где $i = \overline{1,4}$, равняется $(w^2–w)/2=(4^2–4)/2=6$. Это означает, что шесть элементов матрицы модифицированных направляющих чисел, находящихся под главной диагональю, могут принимать произвольные двоичные значения. Количество подобных матриц и соответственно множеств модифицированных направляющих чисел равняется $2^{(w^2-w)/2} = 2^6 = 64.$ Две из указанных матриц V_1 и V_2 приводились ранее.

Максимально возможное количество уникальных множеств из w модифицированных направляющих чисел μ_i равняется $2^{(w^2-w)/2}$, а их элементы могут формироваться с использованием различных алгоритмов генерирования равновероятных двоичных цифр. В случае использования для этих целей M-последовательностей единственным ограничением на степень m порождающего полинома $\varphi(x)$ является равенство $\deg\varphi(x)=(w^2–w)/2$. Выполнение данного равенства обеспечивает формирование различных вариантов значений $\beta_{–j}(i) \in \{0,1\}$ при $j<i$ и соответственно всевозможных последовательностей Соболя для заданного значения w. В случае когда

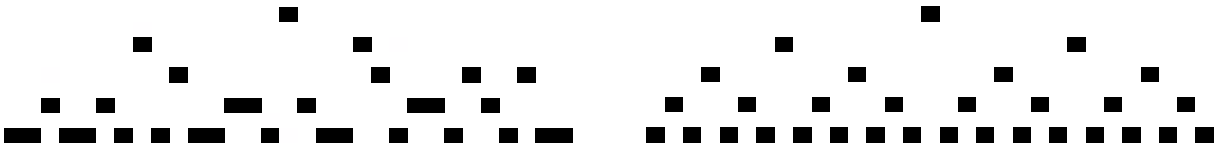
$w = 4$, для порождающего полинома $\varphi(x)$ необходимо, чтобы $\deg\varphi(x) = 6$, что позволит получить все возможные матрицы вида V_1 и V_2, которые будут отличаться значениями элементов, находящихся ниже главной диагонали.

6. Применение последовательностей Соболя в качестве тестовых воздействий

Основой для применения последовательностей Соболя в качестве тестовых воздействий служит их «более равномерное» распределение по сравнению с псевдослучайными последовательностями, которые являются базовыми для реализации практически всех современных методов тестирования вычислительных систем [5, 7, 9, 11, 12, 18, 22–26].

Распределение последовательных псевдослучайных значений (рис. 6, *а*), полученных на основании полинома $\varphi(x)=1\oplus x^2\oplus x^5$, и значений последовательности Соболя (рис. 6, *б*), представленных в табл. 5, характеризуется различной степенью равномерности.

На рис. 6 показана временная последовательность генерирования псевдослучайных значений {16, 8, 20, 10, 21, 26, 29, 14, 23, 27, 13, 6, 3, 17, 24, 28, 30, 31, 15, 7, 19, 25, 12, 22, 11, 5, 18, 9, 4, 2, 1} и элементов последовательности Соболя {16, 8, 24, 12, 28, 4, 20, 18, 2, 26, 10, 30, 14, 22, 6, 27, 11, 19, 3, 23, 7, 31, 15, 9, 25, 1, 17, 5, 21, 13, 29} на интервале целых чисел от 1 до 31 в зависимости от длины последовательностей 2^k-1, $k = \overline{1, w}$. Из рис. 6 видно, что для $k = 3$ последовательность значений 16, 8, 20, 10, 21, 26, 29 из $2^3-1 = 7$ псевдослучайных чисел распределена «менее равномерно» по сравнению с числами Соболя 16, 8, 24, 12, 28, 4, 20. В общем случае указанное свойство «большей равномерности» выполняется для всех значений k и пространств различной размерности [27, 33, 34].



а) *б)*

Рис. 6. Примеры последовательностей: *а)* псевдослучайной; *б)* Соболя

Структура квазислучайных последовательностей Соболя максимально равномерно распределяет тестовые воздействия по всему пространству входных наборов, что позволит достичь большей их покрывающей способности по отношению к типовым входным тестовым наборам (рис. 3 и 4). Напомним, что, как и в известных методах тестирования вычислительных систем [5, 7, 9, 11, 12, 18, 22–26], в данном случае также используется принцип черного ящика. Факт максимально равномерного распределения тестовых воздействий позволяет увеличить значение традиционных мер оценки качества тестов, а именно вероятности обнаружения тестом хотя бы одной неисправности (P-measure) и вероятности обнаружения ожидаемого количества неисправностей (E-measure) [21–24].

В качестве экспериментального подтверждения эффективности кавазислучайного тестирования вычислительных систем в рамках данной статьи были проведены исследования применимости данного подхода для тестирования запоминающих устройств. Исследовались запоминающие устройства, состоящие из матрицы 1024×1024 запоминающих ячеек. В качестве их неисправностей рассматривались многократные неисправности, представляющие собой блоки из 2×2, 3×3 и 4×4 ячеек, содержащие константные неисправности. В качестве теста использовался MATS+, который характеризуется 100%-й полнотой обнаружения константных неисправностей. Рассматривались два случая формирования адресных последовательностей, а именно псевдослучайные адресные последовательности и квазислучайные двухмерные последовательности Соболя. Отмечено существенное уменьшение времени тестирования до обнаружения факта неисправного поведения запоминающего устройства. Во всех случаях длина теста в среднем уменьшается на величины от 44 до 53 % по отношению к псевдослучайным адресным последовательностям. Данные исследования подтверждают результаты,

ранее полученные для программных приложений. Так же, как и в случае управляемых вероятностных тестов [24–26], квазислучайные тесты позволяют существенно уменьшить количество тестовых наборов для обнаружения первой неисправности (F-measure) в программном продукте [21–23]. Результаты, приведенные в [31], свидетельствуют о 50 %-м улучшении значений данной метрики даже в случае использования простейших классических квазислучайных последовательностей.

Заключение

В статье рассмотрена причинно-следственная зависимость между неисправными состояниями вычислительных систем и типовыми тестовыми воздействиями для их обнаружения. Показана идентичность типовых тестовых воздействий как для программной, так и для аппаратной части вычислительных систем, что позволяет использовать единую методологию для тестирования вычислительных систем в целом. В качестве последовательностей тестовых воздействий вычислительных систем, адекватно покрывающих множество типовых входных значений (рис. 3 и 4), обосновывается использование квазислучайных последовательностей. Основное внимание уделено последовательностям Соболя как наиболее эффективным для формирования бинарных тестовых воздействий. Проведенный анализ последовательностей Соболя позволил предложить ряд модификаций, позволяющих существенно увеличить их количество.

Список литературы

1. Rajsuman, R. System-On-A-Chip: Design and Test / R. Rajsuman. – London : Artech House Publishers, 2000. – 303 p.

2. Иванюк, А.А. Проектирование встраиваемых цифровых устройств и систем / А.А. Иванюк. – Минск : Бестпринт, 2012. – 338 с.

3. Semiconductor Intellectual Property: Continuing on the Path to Grow. ASIC IP report / Semiconductor Research Corp. Research Triangle Park. – NC, USA, 2007. – 100 p.

4. Harris, I.G. Hardware-Software Co-validation: Fault Models and Test Generation / I.G. Harris // IEEE Design & Test of Computers. – 2003. – Vol. 20, № 1. – P. 40–47.

5. Hamill, M. Common Trends in Software Fault and Failure Data / M. Hamill, K. Goseva-Popstojanova // IEEE Transaction on Software Engineering. – 2009. – Vol. 35, № 4. – P. 484–496.

6. Comparative Analysis of Hardware and Software Fault Tolerance: Impact on Software Reliability Engineering / H. Ammar [et al.] // Institute for Software Research Fairmont, WV 26554 USA [Electronic resourse] January 15, 1999. – Mode of access : www.isr.wvu.edu. – Date of access : 12.07.2012.

7. Ярмолик, С.В. Маршевые тесты для самотестирования ОЗУ / С.В. Ярмолик, А.П. Занкович, А.А. Иванюк. – Минск : Изд. центр БГУ, 2009. – 270 с.

8. White, L. A domain strategy for computer program testing / L. White, E. Cohen // IEEE Transaction on Software Engineering. – 1980. – Vol. SE–6, № 3. – P. 247–257.

9. Schnekenburger, C. Towards the determination of typical failure patterns / C. Schnekenburger, J. Mayer // In Proc. of Fourth Intern. Workshop on Software Quality Assurance: in conjunction with the 6th ESE/FSE joint meeting, ser. SOQUA'07. – N.Y., USA : ACM, 2007. – P. 90–93.

10. Proportional sampling strategy guidelines for software testing practitioner / F.T. Chan [et al.] // Information and Software Technology – 1996. – Vol. 38, № 12. – P. 775–782.

11. Shahbazi, A. Centroidal Voronoi Tessellation – a New Approach to Random Testing / A. Shahbazi, A.F. Tappenden, J. Miller // IEEE Transaction on Software Engineering. – 2013. – Vol. 39, № 2. – P. 163–183.

12. Fault Pattern Oriented Defect Diagnosis for Memories / C.W. Wang [et al.] // In Proc. of Intern. Test Conference (ITC 2003). – Charlotte, NC, USA, 2003. – P. 29–38.

13. Using electrical bitmap results from embedded memory to enhance yield / A. Segal [et al.] // IEEE Design & Test of Computers. – 2001. – Vol. 15, № 3. – P. 28–39.

14. Goor, A.J. Testing Semiconductor Memories, Theory and Practice / A.J. Goor. – UK, Chichester : John Wiley & Sons, 1991. – 536 p.

15. Barzilai, Z. Exhaustive Generation of Bit Pattern with Application to VLSI Self–Testing / Z. Barzilai, D. Coppersmith, A. Rozenberg // IEEE Transactions on Computers. – 1983. – Vol. C–31, № 2. – P. 190–194.

16. Furuya, K. A probabilistic approach to locally exhaustive testing / K. Furuya // IEEE Transactions on IEICE. – 1989. – Vol. E72, № 5. – P. 656–660.

17. Testing of embedded RAM using exhaustive random sequence / H. Maeno [et al.] // In Proc. of Intern. Test Conference (ITC 1987). – Washington, D.C., USA, 1987. – P. 105–110.

18. Mrozek, I. Antirandom Test Vector for BIST in Hardware/Software Systems / I. Mrozek, V. Yarmolik // Fundamenta Informaticae. – 2012. – Vol. 119, № 2. – P. 163–185.

19. Malaiya, Y.K. The coverage problem for random testing / Y.K. Malaiya, S. Yang // In Proc. of Intern. Test Conference (ITC 1984). – Philadelphia, PA, USA, 1984. – P. 237–242.

20. Malaiya, Y.K. An examination of fault exposure ratio / Y.K. Malaiya, A. Mayrhauser, P.K. Srimani // IEEE Transactions on Software Engineering. – 1993. – Vol. 19, № 11. – P. 1087–1094.

21. Malaiya, Y.K. Antirandom Testing: Getting the most out of Back-Box Testing / Y.K. Malaiya, S. Yang // In Proc. of Sixth Intern. Symposium on Software Reliability Engineering. – Toulouse, France, 1995. – P. 86–95.

22. Antirandom Testing: A Distance-Based Approach / S.H. Wu [et al.] // VLSI Design. – 2008. – № 2. – P. 1–9.

23. Fast Antirandom (FAR) Test Generation / A. Mayrhause [et al.] // In Proc. of the Third IEEE Intern. High-Assurance System Engineering Symposium. – Washington, D.C., USA, 1998. – P. 262–269.

24. Mrozek, I. Iterative Antirandom Testing / I. Mrozek, V.N. Yarmolik // Journal of Electronic Testing: Theory and Applications (JETTA). – 2012. – Vol. 9, № 3. – P. 251–266.

25. Ярмолик, С.В. Управляемое случайное тестирование / С.В. Ярмолик, В.Н. Ярмолик // Информатика. – 2011. – № 1(29). – С. 79–88.

26. Ярмолик, С.В. Управляемые вероятностные тесты / С.В. Ярмолик, В.Н. Ярмолик // Автоматика и телемеханика. – 2012. – № 10. – С. 142–155.

27. Sobol, I.M. Uniformly distributed sequences with an additional uniform property / I.M. Sobol // USSR Comput. Math. Math. Phys. – 1976. – Vol. 16. – P. 236–242.

28. Niederreiter, H. Point sets and sequences with small discrepancy / H. Niederreiter // Monatshefte fur Mathematik. – 1987. – Vol. 104, № 4. – P. 273–337.

29. Halton, J.H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals / J.H. Halton // Numerische Mathemati. – 1960. – Vol. 2, № 1. – P. 84–90.

30. Chi, H. Computational investigations of quasi-random sequences in generating test cases for specification-based tests / H. Chi, E.I. Jones // Proc. of the 38th on Winter Conference, ser. WSC'06. Winter Simulation Conference. – Monterey, CA, USA, 2006. – P. 975–980.

31. Chen, T.Y. Quasi-Random Testing / T.Y. Chen, R. Merkel // IEEE Transaction on Reliability. – 2007. – Vol. 56, № 3. – P. 562–568.

32. Ярмолик, В.Н. Генерирование и применение псевдослучайных сигналов в системах испытаний и контроля / В.Н. Ярмолик, С.Н. Демиденко. – Минск : Наука и техника, 1986. – 200 с.

33. Соболь, И.М. Вычисление несобственных интегралов при помощи равномерно распределенных последовательностей / И.М. Соболь // Доклады АН СССР. – 1973. – Т. 210, № 2. – С. 278–281.

34. Соболь, И.М. Точки, равномерно заполняющие многомерный куб / И.М. Соболь. – М. : Знание, 1985. – 32 с.

35. Savage, C. A survey of combinatorial Gray code // SIAM Review / C. Savage. – 1997. – Vol. 39, № 4. – P. 605–629.

Поступила 02.06.13

Белорусский государственный университет
информатики и радиоэлектроники,
Минск, П. Бровки, 6
e-mail: yarmolik@cosmostv.by,
yarmolik10ru@yahoo.com

S.V. Yarmolik, V.N. Yarmolik

QUASI-RANDOM TESTING OF COMPUTER SYSTEMS

Various modified random testing approaches have been proposed for computer system testing in the black box environment. Their effectiveness has been evaluated on the typical failure patterns by employing three measures, namely, P-measure, E-measure and F-measure. A quasi-random testing, being a modified version of the random testing, has been proposed and analyzed. The quasi-random Sobol sequences and modified Sobol sequences are used as the test patterns. Some new methods for Sobol sequence generation have been proposed and analyzed.