ЛИТЕРАТУРА

1. Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomir Mech. L-systems: from the theory to visual models of plants. In M. T. Michalewicz, editor, *Proceedings of the 2nd CSIRO Symposium on Computational Challanges in Life Sciences*. CSIRO Publishing, 1996.

2. Radomir Mech and Przemyslaw Prusinkiewicz. Visual Models of Plants Interacting with Their Environment. Proceedings of SIGGRAPH 96 (New Orleans, Louisiana, August 4-9, 1996). In *Computer Graphics* Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH. P. 397–410.

3. Przemyslaw Prusinkiewicz, Mark James, and Radomir Mech. Synthetic Topiary. Proceedings of SIGGRAPH 94 (Orlando, Florida, July 24-29, 1994). In *Computer Graphics* Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH. P. 351–358.

4. Микуцкий В.С. Агрегированная модель динамики древостоя. – Мн.,1996. – 18 с. (Препринт / Ин-т техн.кибернетики АН Беларуси; N 1).

5. Бадд Т. Объектно-ориентированное программирование в действии / Пер. с англ. – СПб.: Питер, 1997. – 464 с.

6. Чернавский Д. С. Синергетика и информация: Динамическая теория информации. – М.:Наука, 2001. – 244 с.

УДК 621.3.049.77

А.В. Орлов, ассистент; Д.М. Романенко, аспирант

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РАСПРЕДЕЛЕНИЯ И ИСПРАВЛЕНИЯ ОШИБОК В ЭЛЕМЕНТАХ ПАМЯТИ

The article are considered some aspects of modeling memory, failure distribution, optimization of correction the mistakes with a new type of the code – three-dimensional iterative code. New model of the memory is analyzed.

Компьютерная модель памяти предназначена для рассмотрения алгоритмов распределения отказов в элементах памяти, оптимизации нахождения ошибок, расчета и определения эффективности исправляющих корректирующих кодов. Структурная схема памяти (рис. 1) представлена в виде изменяющего набора микросхем памяти с плавающими характеристиками, т. е. количество и размерность можно задавать вручную. Для этого используется диалоговое окно, где задается количество микросхем памяти, количество строк и столбцов (учитывая столбцы и строки исправляющих кодов) в каждой микросхеме.

Максимальное количество микросхем памяти – $7FF0_{16}$ (32752), максимальное количество строк и столбцов (с учетом исправляющих кодов) по $7FF0_{16}$ (32752).

Анализ методом имитационного моделирования заключается в определении происходящих в системе памяти событий и их привязки к моментам дискретного времени и элементам системы.

Время, в котором функционирует моделируемый объект, называется *модельным*. Дискретизация модельного времени производится путем разбиения заданного для имитации интервала $[0, T_k]$ на участки. Любое событие, происшедшее на участке с границами t_i и t_{i+1}, считается происшедшим в момент времени t_{i+1}.

Рис. 1. Структурная схема элементов памяти

Математическая модель, готовая к имитации прохождения заявок, представляет собой программу, включающую ряд операторов. Операторы, относящиеся к группе основных, вызывают действия над заявками. Это операторы генерирования заявок, направления заявок к определенным узлам (микросхема памяти, строка, столбец, ячейка), постановки в очередь на обслуживание, занятия узла, освобождения и др. Исполнение такой программы по отношению к конкретной заявке выявляет ее маршрут и время прохождения этапов маршрута. Однако поведение заявки в системе не является независимым, оно обусловливается событиями, в которых фигурируют также другие заявки. Поэтому процесс имитации на ЭВМ должен отображать хронологию событий в последовательности, имеющей место в реальной работе микросхем памяти. В связи с этим исполнение программы имитации по отношению к конкретной заявке, например заявке ***A***, должно прерываться, когда заявка задерживается в очереди или в обслуживающем аппарате, и программа должна переходить к имитации более ранних событий, чем события, связанные с дальнейшим продвижением по маршруту заявки ***A***.

Для соблюдения при имитации правильной хронологии событий все события, которые в данный момент модельного времени уже можно предвидеть, упорядочиваются по времени наступления и оформляются в виде списка. Элементами списка являются имена (ссылки) заявок, с которыми связаны рассматриваемые события. В списке событий можно выделить его начало — *список текущих событий,* т. е. событий, которые должны произойти в текущий момент времени t_i. Остальная часть списка называется *списком будущих событий.*

После того как программа прерывает имитацию прохождения заявкой ***A*** своего маршрута, она переходит к обработке очередного события из списка текущих событий. Пусть это событие связано с заявкой ***B***. Программа исполняется, начиная с того оператора, который ранее определил задержку заявки ***B***, и доходит до такого оператора, который означает новую задержку заявки ***B***. Если эта задержка происходит в связи с обслуживанием в некотором устройстве, то при этом определяется длительность обслуживания и, следовательно, момент времени $t_i + \tau$, в который обслуживание закончится. Далее ссылка на заявку ***B*** включается в список будущих событий на место, определяемое значением $t_i + \tau$. После этого программа выбирает очередной элемент из списка текущих событий и имитирует продвижение новой заявки. Сведения о том месте, в кото-

ром была задержана заявка и с которого, следовательно, нужно возобновлять продвижение, фиксируются в специальном массиве параметров заявок.

После того как все события текущего момента времени t_i исчерпаны, модельное время увеличивается до значения времени t_{i+1} ближайшего события, список событий в момент времени t_{i+1} становится списком текущих событий. Процесс имитации продолжается по тому же алгоритму. Окончание процесса имитации связывается с исчерпанием всего списка событий или с прохождением всего заданного интервала времени T_k.

Следовательно, имитационное моделирование осуществляется путем многократного просмотра операторов имитационной модели и выполнения предписываемых ими действий. Просмотр операторов относится только к тем моментам времени, в которые произошли какие-либо события. При просмотре происходят обращения к математическим моделям тех элементов памяти, с которыми связаны происходящие события. Такое моделирование, избирательное по времени и месту событий, называется *событийным.* Использование принципа событийности позволяет выполнять только те действия, которые действительно необходимы для получения результата, и, следовательно, приводит к сокращению затрат машинного времени.

Программная реализация алгоритмов событийного моделирования представляет собой пакет подпрограмм, реализующий математические модели элементов, программы, осуществляющие процедуры выработки случайных чисел, статистической обработки результатов моделирования и др. Управляющая программа пакета содержит обращения к подпрограммам устройств, источников, памятей, с ее помощью реализуются ввод исходных данных, упорядочение списков событий, изменение модельного времени, продвижение заявок.

В базе данных (БД), с которой работает программа имитации, имеются следующие основные массивы данных.

Массив очередей и списков. Массив очередей упорядочен по устройствам, массив списков – по моментам наступления событий. Элементами этих массивов являются ссылки на заявки. Поэтому при обработке списков по этим элементам могут быть найдены адреса, по которым хранятся сведения о самих заявках.

Массив заявок. Здесь хранятся сведения о каждой заявке: имя заявки, ее тип, значения приоритетов, время очередного события, связанного с заявкой, текущее место нахождения заявки. Доступ к этому массиву осуществляется по ссылкам заявок, содержащимся в массивах очередей и списков.

Массив параметров элементов. В нем содержатся значения таких параметров, как числовые характеристики законов распределения интервалов обслуживания, данные о дисциплинах обслуживания, накопленные суммы, характеризующие на текущий момент время занятого состояния каждого устройства, количество обслуженных заявок, длины очередей, среднее время ожидания в очередях и т. п.

Процесс исполнения программы имитации и ее взаимодействия с БД показан на рис. 2.

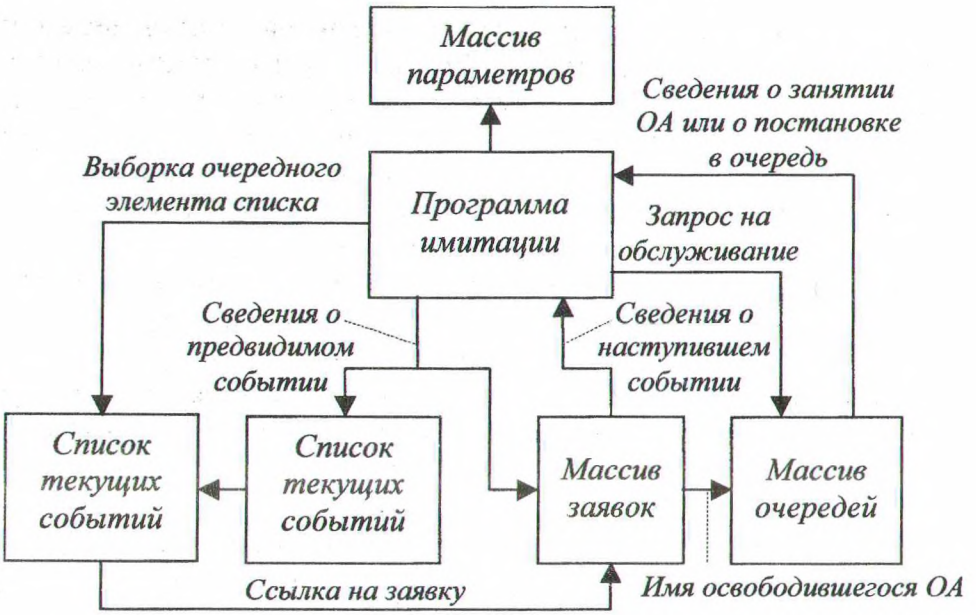

Рис. 2. Взаимодействие программы имитации с массивами БД

Каждый этап вычислений начинается с выборки очередного элемента из списка текущих событий. Этот элемент представляет собой имя заявки (ее адрес в массиве заявок). Обозначим имя обрабатываемой заявки ***A***. По извлеченным из массива заявок сведениям определяется имя освобождающегося ***ОА*** и номер оператора в программе имитации, с которого будет прослеживаться продвижение заявки ***A***. Сначала программа имитации проверяет наличие очереди на входе освободившегося ***ОА***, и, если очередь имеется, ***ОА*** начинает обслуживать некоторую заявку ***B*** из очереди в соответствии с заданной дисциплиной обслуживания. При этом происходит обращение к подпрограмме рассматриваемого ***ОА***, определяется время обслуживания и тем самым становится предвидимым новое событие – окончание обслуживания заявки ***B*** в этом ***ОА***. Ссылка на это событие помещается в список будущих событий, и производятся изменения сведений о заявке ***B*** в массиве заявок. Далее имитируется продвижение в системе заявки ***A***. Это продвижение прерывается, когда в соответствии с заданной программой производится запрос на обслуживание заявки ***A*** в некотором ***ОА***. При этом происходит обращение к массиву очередей по имени этого ***ОА*** и либо поступление заявки ***A*** на обслуживание, либо ее постановка в очередь. Изменения, происшедшие в состояниях очередей, обслуживающих аппаратов и заявок, отражаются в изменении соответствующих данных в массиве параметров.

В качестве корректирующих кодов предлагается использовать новые виды итеративных кодов:

- двухмерный линейный итеративный код с диагональными проверками (позволяет корректировать ошибки кратностью не выше 2);
- трехмерный линейный итеративный код (кратность корректируемых ошибок – не выше 3);
- трехмерный линейный итеративный код с проверочными символами по диагонали (кратность корректируемых ошибок – не выше 4).

Основные свойства данных кодов приведены в [5]. Как видно, коды отличаются друг от друга кратностью корректируемых ошибок, а следовательно, и избыточностью.

Для систем памяти с интеграцией на полупроводниковой пластине целесообразно использовать трехмерный линейный итеративный код с проверочными символами по диагонали. Это объясняется схожестью его структуры со структурой организации памяти, высокой кратностью корректируемых ошибок, простотой алгоритмов коррекции. Поэтому рассмотрим более подробно алгоритмы коррекции для данного кода.

Минимальное кодовое расстояние данного вида кода, согласно [4,5], будет d=10, а это значит, что код исправляет до четырех ошибок включительно.

Известно, что декодирование любого кода является наиболее трудоемкой операцией. Сложности реализации декодеров особенно возрастают при коррекции ошибок, число которых больше единицы. При этом иногда возникают ситуации, когда сложность декодера приводит к таким значительным временным и аппаратурным затратам, что сводит на нет корректирующие способности кода. Авторами разработан специальный алгоритм коррекции ошибок кратностью не выше четырех. Данный алгоритм строится на следующих принципах :

1) идентификации многократных ошибок: по специально разработанным критериям и признакам определяют тип ошибки по **hvdz** – классификации [5];
2) по типу ошибки выбирается соответствующий алгоритм коррекции;
3) реализуется соответствующий алгоритм коррекции.

Если ограничить кратность ошибок, которые должны быть исправлены, до (d – 1) / 2 + 1 (если d – нечетное) и до (d – 2) / 2 + 1 (если d – четное), где d – минимальное кодовое расстояние трехмерного итеративного кода с проверочными символами по диагонали, то для коррекции ошибок можно использовать следующий алгоритм: в случае возникновения ошибки кратностью (d – 1) / 2 + 1 или (d – 2) / 2 + 1 координаты ошибки определяются по Z – рядам, "показавшим" о наличии ошибки; в случае возникновения ошибок кратностью меньше (d – 1) / 2 + 1 или (d – 2) / 2 + 1 ошибки исправляются "силами" линейных итеративных кодов с проверочными символами по диагонали. Для реализации данного алгоритма можно использовать урезанную проверочную матрицу, т.е. из проверочной матрицы исключаются паритеты паритетов. В данном случае, хотя и происходит уменьшение кратности корректируемых ошибок, одновременно снижается избыточность кода, а также существенно упрощается алгоритм коррекции. Так, избыточность стандартной проверочной матрицы трехмерного итеративного кода с проверочными символами по диагонали при k = 1024 равна r = 560; при той же длине информационного слова для усеченной проверочной матрицы r = 513. Избыточность уменьшилась на 8.4%.

Данный алгоритм является общим для всех трехмерных кодов, полученных путем прямого (кронекеровского) произведения двух кодов, при условии, что одним из кодов–сомножителей является свертка по модулю два. Вторым сомножителем может быть любой код, например код Хемминга.

ЛИТЕРАТУРА

1. Норенков И.П., Маничев В.Б. Системы автоматического проектирования электронной и вычислительной аппаратуры. – М.: Высшая школа, 1983. – 272 с.

2. Северцев Н.А. Надежность сложных систем в эксплуатации и отработке. – М.: Высшая школа, 1989. – 432 с.

3. Урбанович П.П., Алексеев В.Ф., Верниковский Е.А. Избыточность в полупроводниковых интегральных микросхемах памяти. – Мн.: Навука і тэхніка, 1995. – 262 с.

4. Мак-Вильямс Ф., Слоэн Н. Теория кодов, исправляющих ошибки / Пер. с англ. под ред. Л.А. Басалыго.–М:Связь, 1979. – 746 с.

5. Урбанович П.П., Романенко Д.М. Свойства и алгоритмы аппаратной реализации нового вида итеративных кодов для систем памяти // Новые информационные технологии: Третья международная конференция NITE'2000. Т. 2. – Мн.: БГЭУ, 2000. – С. 159–164.

УДК 621.3.049.77

А.И. Галабурда, аспирант

МОДЕЛИРОВАНИЕ ПАРАМЕТРОВ НАДЕЖНОСТИ УСТРОЙСТВ КРИПТОГРАФИЧЕСКОГО ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ

The main idea of this article – to estimate failure effect of symmetric data encryption algorithms hardware. Were established the most error prone blocks of block chipper algorithms. New methods of improvements of chipper blocks are suggested to enhance stability.

Широкое применение компьютерных технологий в автоматизированных системах обработки информации и управления привело к обострению проблемы защиты информации, циркулирующей в компьютерных системах, от несанкционированного доступа. Радикальное решение проблемы защиты информации может быть получено только на базе использования криптографических методов. При внедрении оборудования, реализующего алгоритмы криптографического преобразования информации, встает вопрос его отказоустойчивости. Данной проблеме и посвящена эта статья.

Сложность схем, разнообразие видов отказов приводят к тому, что аналитические методы расчета надежности схем криптографического преобразования информации обеспечивают лишь приближенную оценку. Метод статистического моделирования надежности позволяет при достаточной длительности моделирования получить более точную, по сравнению с аналитическими методами, оценку показателей надежности с помощью более точных моделей отказов.

Построим дескриптивную модель криптоалгоритма ГОСТ 28147-89 и проанализируем один из вариантов его аппаратной реализации на предмет устойчивости к сбоям оборудования. Сбои оборудования являются следствием сложных физических процессов, происходящих в аппаратуре преобразования данных, а именно в таких компонентах криптосхем, как логические, арифметические элементы, элементы памяти и др., могут появляться [1] статистически независимые отказы и сбои. Следствием влияния сбоев и отказов в криптографических схемах на достоверность преобразуемой информации будем считать ошибки. Структурная схема простейшего устройства, реализующего шифрование информации в режим простой замены, состоит из следующих блоков:

- два 32-разрядных накопителя N_1 и N_2;
- ключевое запоминающее устройство, состоящее из восьми 32-разрядных накопителей;
- 32-разрядный сумматор по модулю 2^{32};
- 32-разрядный сумматор по модулю 2;
- модуль циклического сдвига на одиннадцать шагов в сторону старшего разряда;
- блок подстановки.