УДК 681.513

М.К. АЛЬ МАДИ

Черкасский государственный технологический университет, Украина

ОСОБЕННОСТИ ПРОГРАММЫ ВЫБОРА ОПТИМИЗИРОВАННОЙ ПОСЛЕДОВАТЕЛЬНОСТИ ТЕСТОВ ДЛЯ ДИАГНОСТИРОВАНИЯ ПОЛУПРОВОДНИКОВОЙ ПАМЯТИ

Приведены особенности программной реализации метода выбора парето-оптимальных тестов для диагностирования полупроводниковых запоминающих устройств. Излагаются алгоритм и принцип построения программы сортировки массива структурированных данных по ключевым признакам объектов.

диагностирование, запоминающие устройства, сортировка данных, алгоритм “пузырька”

Введение

При постоянно увеличивающейся информационной емкости микросхем оперативной памяти актуальной становится проблема обеспечения их высокой надежности и качества, при этом производители стремятся сократить трудоемкость и продолжительность испытаний. Обеспечение допустимого уровня экономичности диагностирования микросхем и модулей оперативной памяти является одной из задач, которую необходимо решать менеджерам отделов технического контроля качества фирм проектирующих, изготавливающих и эксплуатирующих средства вычислительной техники, которые предназначены для построения систем критического применения.

Осуществление эффективного диагностирования изделий полупроводниковой памяти связано с решением задачи определения оптимизированного набора диагностических тестов, обеспечивающих быструю локализацию неисправностей. Каждая фирма самостоятельно решает, какое время тестового диагностирования будет для нее оптимальным и позволит достичь заданного качества изделий. Для уменьшения трудоемкости решения данной задачи необходима разработка специальных методов, алгоритмов и программ, позволяющих использовать широкие возможности современной вычислительной техники для сортировки массивов данных, описывающих априорные свойства применяемых тестов.

В известных алгоритмах сортировки данных обрабатываются массивы целых или вещественных чисел и в результате формируется новый упорядоченный по возрастанию массив чисел [1]. Однако данные алгоритмы после выполнения операции сортировки не позволяют идентифицировать свойства каких объектов оценивают упорядоченные элементы полученного массива.

Целью данной статьи является разработка метода и алгоритма программы сортировки массива структурированных данных по ключевым признакам объектов, которые позволят идентифицировать данные объекты после выполнения операции сортировки.

1. Математическая модель и решение задачи исследования

При выборе оптимизированной последовательности тестов для диагностирования микросхем и модулей памяти необходимо учитывать, что их диагностические свойства нечетко определены, а в ряде случаев предлагаются тесты; для которых не излагается даже их подробное алгоритмическое описание. В условиях наличия нечетких априорных

данных о диагностических свойствах тестов целесообразно применять метод выбора парето-оптимальных тестов, в котором критерии качества тестов определяются квалифицированными специалистами в области диагностирования полупроводниковых запоминающих устройств [2].

Наиболее важным критерием для оценки свойств тестов является их способность обнаруживать наиболее широко проявляющиеся отказы ячеек памяти, дешифраторов адреса, усилителей считывания, схем восстановления зарядов в запоминающих конденсаторах и другие.

Данные свойства тестов оцениваются вероятностями обнаружения отказов заданных типов и измеряются вещественными числами в диапазоне от 0 до 1 и по этому легко сравниваются.

Вектор сравнения $S_i^{\vartheta\mu}$ свойств тестов ϑ и μ по обнаружению отказов *i*-го типа определяется следующим выражением:

$$S_i^{\vartheta\mu} = 1, если\ q_{i_i}^{\vartheta} > q_{i_i}^{\mu};$$
$$S_i^{\vartheta\mu} = 0, если\ q_{i_i}^{\vartheta} = q_{i_i}^{\mu};$$
$$S_i^{\vartheta\mu} = -1, если\ q_{i_i}^{\vartheta} < q_{i_i}^{\mu},$$

где $q_{i_i}^{\vartheta}$, $q_{i_i}^{\mu}$ - вероятности обнаружения отказов *i*-го вида тестами ϑ и μ соответственно.

Важным фактором является также время выполнения тестов, которое может в зависимости от сложности теста измеряться в микросекундах, часах и даже достигать астрономических значений [3].

Логично, что тесты, продолжительность которых не соответствует экономическим возможностям производства или условиям эксплуатации $t_j > t_{\max}$, следует исключить из рассматриваемого множества.

Для оставшейся группы тестов значение параметра время тестирования следует нормировать и для каждого теста получить вектор оценки свойств по продолжительности выполнения согласно выражению:

$$S_t^j = \frac{\sum_{j=1}^{n} t_j / k}{t_j};$$

где S_t^j - оценка свойств *j*-го теста по времени выполнения;

k - число тестов;

t_j - время выполнения *j*-го теста.

Аналогичным образом можно получить нормированные значения для всех оценок свойств тестов по заданным критериям и представить их в виде табл. 1.

Таблица 1

Оценки свойств тестов для заданных критериев

Название теста	Оценки для критериев				
	K_1	K_2	K_3	•••	K_m
Тест 1	E_{11}	E_{12}	E_{13}	•••	E_{1m}
Тест 2	E_{12}	E_{22}	E_{23}	•••	E_{2m}
•••	•••				
Тест k	E_{k1}	E_{k2}	E_{k3}	•••	E_{km}

Для каждого критерия можно получить его среднее значение по формуле:

$$E_{cp}^j = \sum_{i=1}^{n} E_{ij} / k\ .$$

Тогда нормированное значение оценки C_{ij} свойств *i*-го теста по обнаружению неисправности *j*-го вида можно получить при помощи выражения:

$$C_{ij} = 1,\ \ если\ E_{ij} > E_{cp}^j;$$
$$C_{ij} = 0,\ \ если\ E_{ij} = E_{cp}^j;$$
$$C_{ij} = -1,\ если\ E_{ij} < E_{cp}^j.$$

Обобщенную оценку C_n^i для каждого теста можно получить по формуле:

$$C_n^i = \sum_{j=1}^{n} C_{ij}.$$

Таким образом, задача выбора оптимизированного набора тестов сводится к ранжированию тестов, т.е. к сортировке по возрастанию их обобщенных оценок.

Для программной реализации задачи исследования предлагается использовать определенные пользователем структурированные данные следующего вида:

```
Typedef struct
{
        Int Estimation;
        Int Number;
        Char Name [6];
}       DATA;
```

В качестве входных данных в память компьютера заносится массив структурированных данных, содержащий обобщенные оценки и наименования тестов.

Для сортировки массива данных предлагается использовать алгоритм “пузырька”, когда записи с ”легкими” значениями ключевого поля всплывают вверх наподобие пузырька.

С целью уменьшения шагов алгоритма введен специальный флаг, разрешающий сравнение элементов массива, и в начале данному флагу присваиваем значение единица. После активизации операции сравнения флагу присваиваем значение нуль, а затем попарно сравниваем элементы массива: первый со вторым, второй с третьим и т.д., одновременно сортируем их.

Если перестановка элементов осуществлялась, то флагу снова присваивается значение единица и начинается новый цикл сортировки элементов. Если флаг не установился в единичное состояние, то это означает, что сортировать больше нечего и процесс останавливается.

Продолжительность сортировки массива данных по данному алгоритму в худшем случае составляет $n!$ операций. Таким образом, при сортировке анализируются только ключевые поля, содержащие обобщенные оценки свойств тестов, а переставляются все сведения о тесте, что позволяет идентифицировать номера и имена ранжированных тестов и определить парето-оптимальную последовательность тестов.

Блок схема алгоритма сортировки массива структурированных данных приведена на рис. 1.

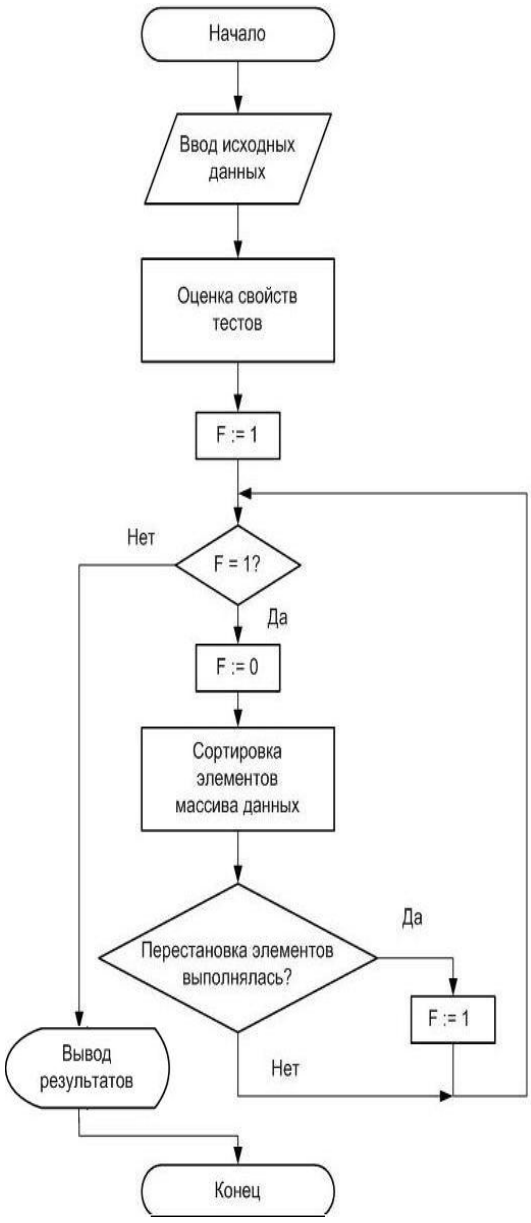

Рис. 1. Алгоритм сортировки массива данных

Программа Optimal_Test, осуществляющая выбор парето-оптимальных тестов, разработана на языке Borland C++, занимает 4.2 кбайт исходного текста и 865 кбайт исполняемого модуля.

Проблема применения символов русского алфавита в тексте программы на языке C++ решена применением специального следующего исполняемого bat файла:

```
C:\borland\bin\bc.exe
C:\borland\bin\keyrus.com
```

Программа `keyrus.com` позволяет применять в исходном тексте символы русского языка, а переключение типов шрифтов производится при помощи комбинации клавиш левой и правой shift следующим образом: левый shift - английский язык, правый shift - русский язык.

2. Практическое применение метода для определения парето-оптимального набора тестов

Программа Optimal_Test применялась для выбора парето-оптимального набора из 15 тестов, приведенных в табл. 2 и обеспечивающих локализацию наиболее распространенных отказов микросхем памяти [4].

Таблица 2

Сведения об исследуемых тестах

№	Название теста	Формула для расчета времени	Время выполнения для t_c = 5 нсек и емкости 4 Мбит, сек
1	WalkRow	8n+2nC	86,067
2	WalkColumn	8n+2nR	86,067
3	GalRow	6n+2nC	171,925
4	GalColumn	6n+4nR	171,925
5	Hammer	49n	1,028
6	March SL	41n	0,86
7	March RAW	26n	0,545
8	March SS	22n	0,461
9	March G	23n	0,482
10	March SR	14n	0,294
11	PMOVI	13n	0,273
12	March C	10n	0,21
13	MATS++	6n	0,126
14	MATS+	5n	0,105
15	Scan	4n	0,084

В табл. 2 принято следующие определения символов: R – число строк ячеек памяти, C – число столбцов.

Гистограмма роста продолжительности диагностирования произвольной последовательности тестов для повышенного напряжения электропитания микросхем памяти HVcc приведена на рис. 2.

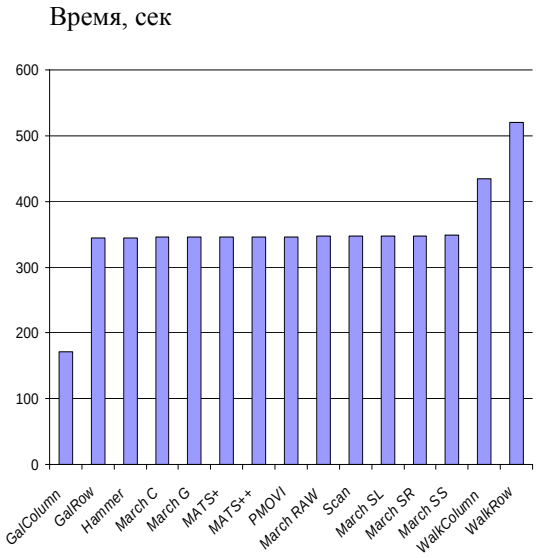

Рис. 2. Продолжительность диагностирования произвольной последовательности тестов

Увеличение продолжительности диагностирования парето-оптимального набора тестов для HVcc отображает гистограмма, приведенная на рис. 3.

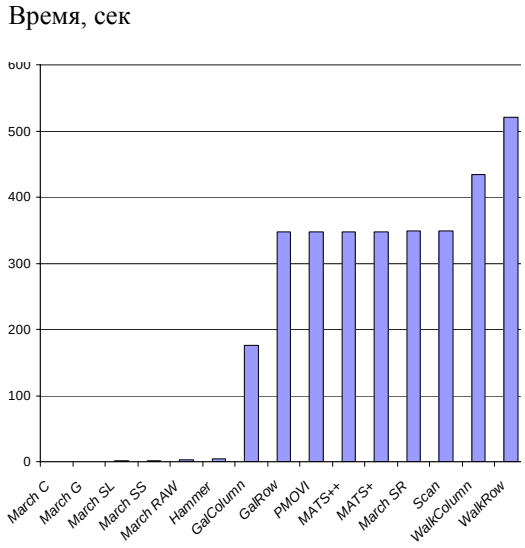

Рис. 3. Продолжительность диагностирования парето-оптимального набора тестов для HVcc

По результатам экспериментальных исследований установлено, что 6 тестов из оптимизированного набора обнаруживают все неисправности в

данной партии микросхем, поэтому применение остальных тестов нецелесообразно. Гистограмма роста продолжительности диагностирования парето-оптимального набора тестов для пониженного напряжения электропитания микросхем памяти LVcc приведена на рис. 4.

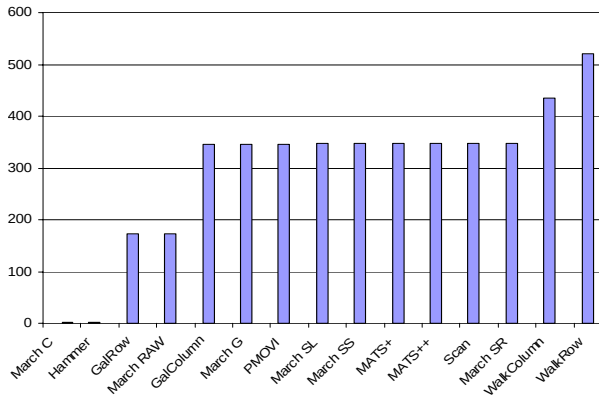

Рис. 4. Продолжительность диагностирования парето-оптимального набора тестов для LVcc

Суммируя продолжительность выполнения 6-ти тестов из произвольной последовательности и парето-оптимального набора, можно определить общие затраты времени на выполнение диагностирования при двух сочетаниях значений напряжений электропитания (рис. 5).

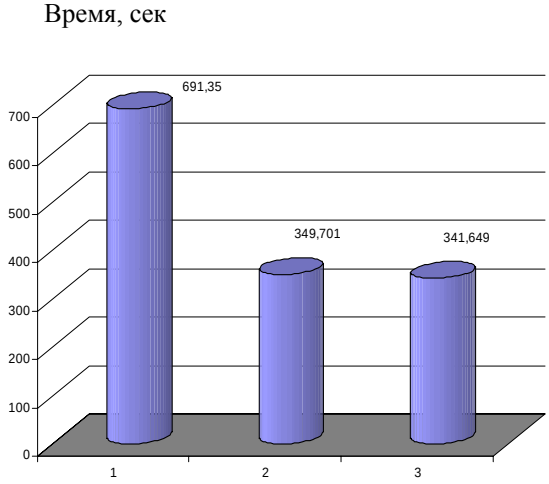

Рис. 5. Сравнение затрат времени

Суммарная продолжительность 6-ти тестов для произвольного набора равна 691,35 сек, для парето-оптимального набора при HVcc и LVcc только 349,701 сек, а экономия времени составляет 341,649 сек.

Выводы

При помощи программы Optimal_Test можно, используя экспериментальные данные о свойствах тестов, определить парето-оптимальный набор тестов, что повышает эффективность диагностирования микросхем памяти, а также надежность вычислительной системы в целом. Программная реализация метода сортировки структурированных массивов данных ”пузырьком” позволяет применять этот алгоритм для решения других важных народнохозяйственных задач, в частности для определения наилучших альтернативных решений при выполнении многоверсионного синтеза цифровых систем критического назначения.

Литература

1. Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы. - М.: Издательский дом ”Вильямс”, 2000. – 384 с.

2. Аль Мади М.К., Андриенко В.А., Рябцев В.Г. Метод выбора парето-оптимальных тестов для диагностирования запоминающих устройств // Радіоелектронні і комп′ютерні системи. – 2005. – № 5 (17). - С. 134-137.

3. Мельников А.В., Рябцев В.Г. Контроль модулей памяти компьютеров. – К.: Корнійчук, 2001. – 172 с.

4. Hamdioui S., Van de Goor Ad J., Reyes D. J., Rodgers M. Memory test experiment: industrial results and data // IEE Proc.-Computer. Digit. Tech. – January 2006. – Vol. 153, No. 1.– P. 1-8.

Поступила в редакцию 29.01.2008

Рецензент: д-р техн. наук, проф. С.М.. Первунинский, Черкасский государственный технологический университет, Черкассы.