УДК 681.327

В. Г. Рябцев, д-р техн. наук
Европейский университет
(Украина, 18008, Черкассы, ул. Смелянская, 83,
тел. 0472630971, E-mail: volodja18@ukr.net),
Д. Н. Моамар
Черкасский государственный технологический университет
(Украина, 18006, Черкассы, бульвар Шевченко, 460,
тел. 0472730271, E-mail: diaamoamar@yahoo.com)

Метод и средство визуализации алгоритмов тестов диагностирования запоминающих устройств

Для синтеза алгоритмов и программ тестов диагностирования запоминающих устройств предложена система визуализации алгоритмов тестов, содержащая устройство управления, квадратную матрицу запоминающих ячеек и четыре головки записи (считывания). Приведен пример синтеза с помощью данной системы теста march C. Сокращение трудоемкости проектных работ обеспечивается динамической визуализацией последовательности выполняемых диагностических операций.

Для синтезу алгоритмів і програм тестів діагностування запам'ятовуючих пристроїв запропоновано систему візуалізації алгоритмів тестів, що містить пристрій управління, квадратну матрицю запам'ятовуючих комірок і чотири головки запису (зчитування). Наведено приклад синтезу за допомогою даної системи тесту march C. Скорочення трудомісткості проектних робіт забезпечується динамічною візуалізацією послідовності виконуваних діагностичних операцій.

К л ю ч е в ы е с л о в а: запоминающее устройство, машина Тьюринга, система визуализации, тест.

Постановка задачи. Современные микросхемы полупроводниковой памяти типа DDR и QDR могут за одну транзакцию сосчитать или записать пакет данных, содержащий несколько слов. При считывании такие микросхемы выдают сначала первое слово, затем второе, третье и так далее или могут изменить последовательность выдачи слов в любой их комбинации. Для выполнения тестового диагностирования современных быстродействующих микросхем памяти целесообразно применять тестеры, имеющие мультипроцессорную структуру, содержащие несколько групп операционных процессоров и обеспечивающие параллельное формирование тестовых воздействий для смежных тактов [1]. Новые функциональные возможности

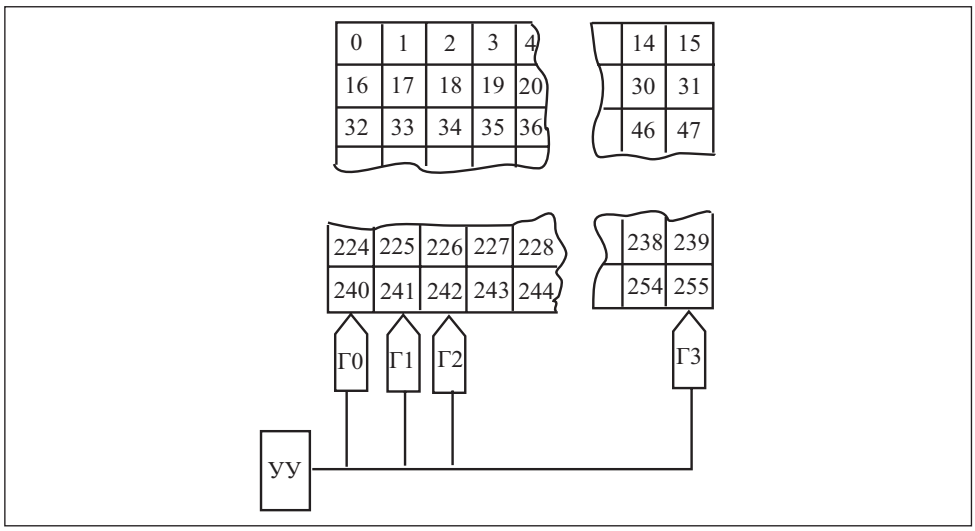

Рис. 1. Структура системы BAT

современных микросхем памяти требуют разработки средств моделирования алгоритмов, способных сократить трудоемкость синтеза программ новых тестов.

Для обеспечения функционирования указанных диагностических устройств распараллеливание алгоритмов тестов необходимо выполнять так, чтобы обеспечить синхронное взаимодействие отдельных операционных процессоров. Для решения данной задачи требуется применение специальных средств визуализации алгоритмов тестов, что позволит существенно снизить трудоемкость проектных работ.

Для формализации понятия алгоритмов используют машины Тьюринга и Поста, позволяющие выделить элементарные операции. Строго выполняя эти операции и наблюдая процесс их выполнения с помощью эмуляторов данных машин, можно получить ожидаемый результат за конечное число шагов [2, 3]. Однако машины Тьюринга и Поста имеют очень ограниченный набор команд, отсутствуют основные арифметические операции и, кроме того, они практически нереализуемы вследствие наличия бесконечных лент, так как память компьютеров ограничена. В этих машинах головку или каретку можно перемещать влево или вправо только на одну позицию, что усложняет моделирование алгоритмов тестов.

Существуют также многоленточные машины Тьюринга, имеющие несколько лент и несколько головок, каждая из которых обозревает только свою ленту, но применяемые ленты бесконечны вправо, поэтому их невозможно реализовать практически.

К недостаткам программирования алгоритмов на вычислительных машинах с архитектурой фон Неймана можно отнести тот факт, что сначала необходимо запрограммировать код адреса, затем код данных и код операции для тестируемого устройства, при этом частота обращения к объекту уменьшается и снижается эффективность тестирования.

Для сокращения трудоемкости синтеза и отладки алгоритмов и программ тестов диагностирования микросхем полупроводниковой памяти предлагается система визуализации алгоритмов тестов (BAT) (рис. 1), которая содержит устройство управления УУ, квадратную матрицу ячеек памяти размером 16 × × 16 и четыре головки записи (считывания) Г0—Г3. Головки могут записывать или считывать из ячеек символы алфавита A и перемещаться по оси X или $У$, или по осям X и $У$ одновременно влево или вправо на 1, 2, ..., p позиций. Перемещения головок независимые, они могут обозревать разные ячейки или одну и ту же ячейку.

Символы, записываемые или считываемые из ячеек памяти, принадлежат алфавиту, который выбирает пользователь. Для матрицы ячеек памяти установлены начальный $a_g = 0$ и конечный $a_n = 255$ адреса, с которых обычно начинается тестирование и заканчивается выполнение теста. Устройство управления системы BAT имеет конечное число состояний, определяемое сложностью реализуемого алгоритма.

Формат команд системы BAT представим в следующем виде:

$$Q_i, S_r^i \to S_r^j, C_r, p_r, Q_j,$$

где Q_i — текущее состояние системы; S_r^i — символ хранившийся в ячейке, которую обозревает r-я головка в текущий момент времени; $r = \overline{0, l-1}$ — номер головки; l — число головок записи (считывания); S_r^j — новый символ, записываемый в ячейку r-й головкой; $C_r \in \{W_r, R_r, A_r\}$ — коды операций для каждой головки; W_r и R_r — операции записи символа в ячейку и считывания из ячейки, сканируемой r-й головкой; A_r — операция сравнения символа, считанного из ячейки, с эталонным значением; p_r — константа изменения позиции r-й головки; Q_j — новое состояние системы.

Алфавит может содержать любой символ с клавиатуры символов:

$$S_r^i = S_r^j = \{U, X, 0, 1, ..., 255, Z\},$$

где U — неопределенное состояние; X — безразличное состояние; Z — символ с клавиатуры, в ячейку матрицы заносится его ASCII-код.

При изменении позиций головками может выполняться одна микрооперация из следующего набора:

$$a_r := a_r + p;\ a_r := a_r - p;\ a_r := a_g;\ a_r := a_n,$$

где p — константа изменения кода адреса. Таким образом, адрес ячейки, которую обозревает головка, может увеличиваться или уменьшаться на p позиций.

Синтез программы теста march C с помощью системы BAT. Рассмотрим реализацию алгоритма наиболее распространенного теста march C [4]. Вначале все головки устанавливаем в положение, когда они обозревают одну и ту же ячейку с начальным адресом, затем перемещаем головку с номером 1 на одну позицию вправо, с номером 2 — на две позиции, с номером 3 — на три позиции:

$$Q_0,\ \ U_r,\rightarrow \forall r, r=\overline{0,3}\ \ a_r;=a_g,\ Q_1,$$

$$Q_1,\ \ U_r,\rightarrow \{a_1:=a_1=1, a_2:=a_2+2, a_3:=a_3+3\},\ Q_2.$$

Записываем фон нулей во все ячейки, которые сканируем с возрастанием кода адреса:

$$Q_2, U_r,\rightarrow \forall r, r=\overline{0,3}\left\{\begin{array}{l}\text{if } a_3\neq a_n \text{ then } W_r(0), a_r:=a_r+4,\ Q_2\\ \text{else } W_r(0), a_r:=a_r+4,\ Q_3\\ \text{end if}\end{array}\right\}.$$

Устанавливаем все головки в исходное положение и перемещаем две головки на одну позицию вправо:

$$Q_3,\ \ 0_r,\rightarrow \forall r, r=\overline{0,3}\ \ a_r;=a_g,\ Q_4,$$

$$Q_4,\ \ 0_r,\rightarrow \{a_2:=a_2+1, a_3:=a_3+1\},\ Q_5.$$

Согласно алгоритму, головки, имеющие номера 0 и 2, считывают символы из ячеек матрицы и сравнивают их с эталонными значениями, а головки с номерами 1 и 3 записывают в ячейки число 255. Затем все головки перемещаются вправо на две позиции:

$$Q_5, 0_r,\rightarrow \left\{\begin{array}{l}\text{if } a_3\neq a_n \text{ then } F_5(Q_5, 0), Q_5\\ \text{else } F_5(Q_5, 0),\ Q_6\\ \text{end if}\end{array}\right\}.$$

Согласно функции $F_5(Q_5,0)$ выполняются следующие действия:

$$F_5(Q_5,0)=\left\{\begin{array}{l}R_0(S_0);\ \text{if } S_0=0 \text{ then } a_0:=a_0+2;\\ \text{else } a_0:=a_0+2,\ Q_{stop};\\ \text{end if;}\\ W_1(255),\ a_1:=a_1+2;\\ R_2(S_2);\ \text{if } S_2=0 \text{ then } a_2:=a_2+2;\\ \text{else } a_2:=a_2+2,\ Q_{stop};\\ \text{end if;}\\ W_3(255),\ a_3:=a_3+2.\end{array}\right.$$

Устанавливаем все головки в положение, когда они обозревают ячейку с начальным адресом, затем две головки с номерами 2 и 3 смещаем на одну позицию вправо:

$$Q_6,\ \ 255_r,\rightarrow \forall r, r=\overline{0,3}\ \ a_r;=a_g\,, Q_7,$$

$$Q_7,\ \ 255_r,\rightarrow \{a_2:=a_2+1, a_3:=a_3+1\}, Q_8.$$

Головки выполняют действия согласно алгоритму теста:

$$Q_8, 255_r,\rightarrow \left\{\begin{array}{l}\text{if } a_3\neq a_n \text{ then } F_8(Q_8,255), Q_8\\ \text{else } F_8(Q_8,255),\ Q_9\\ \text{end if}\end{array}\right\}.$$

В соответствии с функцией $F_8(Q_8,255)$ выполняются следующие действия:

$$F_8(Q_8,255)=\left\{\begin{array}{l}R_0(S_0);\ \text{if } S_0=255 \text{ then } a_0:=a_0+2;\\ \text{else } a_0:=a_0+2,\ Q_{stop};\\ \text{end if;}\\ W_1(0);\ a_1:=a_1+2; R_2(S_2);\\ \text{if } S_2=255 \text{ then } a_2:=a_2+2;\\ \text{else } a_2:=a_2+2,\ Q_{stop};\\ \text{end if;}\\ W_3(0);\ a_3:=a_3+2.\end{array}\right.$$

Для сканирования ячеек с уменьшением кода адреса устанавливаем головки в позиции, когда они обозревают ячейку с конечным адресом:

$$Q_9,\ \ 0_r,\rightarrow \forall r, r=\overline{0,3}\ \ a_r;=a_n\,, Q_{10}.$$

Смещаем две головки с номерами 2 и 3 на одну позицию влево:

$$Q_{10},\ \ 0_r,\rightarrow \{a_2:=a_2-1, a_3:=a_3-1\}, Q_{11}.$$

Головки выполняют действия согласно алгоритму с уменьшением кода адреса:

$$Q_{11}, 0_r,\rightarrow \left\{\begin{array}{l}\text{if } a_3\neq a_g \text{ then } F_{11}(Q_{11},0), Q_{11}\\ \text{else } F_{11}(Q_{11},0),\ Q_{12}\\ \text{end if}\end{array}\right\}.$$

В соответствии с функцией $F_{11}(Q_{11},0)$ выполняются следующие действия:

$$F_{11}(Q_{11},0)=\begin{cases} R_0(S_0);\ \text{if } S_0=0 \text{ then } a_0:=a_0-2; \\ \text{else } a_0:=a_0-2,\ Q_{stop}; \\ \text{end if}; \\ W_1(255),\ a_1:=a_1-2; \\ R_2(S_2);\ \text{if } S_2=0 \text{ then } a_2:=a_2-2; \\ \text{else } a_2:=a_2-2,\ Q_{stop}; \\ \text{end if}; \\ W_3(255),\ a_3:=a_3-2. \end{cases}$$

Снова устанавливаем головки в положения, когда все они обозревают ячейку с конечным адресом, и смещаем головки с номерами 2 и 3 на одну позицию влево:

$$Q_{12},\ \ 255_r,\rightarrow \forall r, r=\overline{0,3}\ \ a_r;=a_n,\ Q_{13},$$

$$Q_{13},\ \ 255_r,\rightarrow \{a_2:=a_2-1, a_3:=a_3-1\},\ Q_{14}.$$

Выполняются действия согласно алгоритму с уменьшением кода адреса:

$$Q_{14},255_r,\rightarrow \left\{\begin{array}{l} \text{if } a_3\neq a_g \text{ then } F_{14}(Q_{14},255),Q_{14} \\ \text{else } F_{14}(Q_{14},255),\ Q_{15} \\ \text{end if} \end{array}\right\}.$$

Согласно функции $F_{14}(Q_{14},255)$ выполняются следующие действия:

$$F_{14}(Q_{14},255)=\begin{cases} R_0(S_0);\ \text{if } S_0=255 \text{ then } a_0:=a_0-2; \\ \text{else } a_0:=a_0-2,\ Q_{stop}; \\ \text{end if}; \\ W_1(0),\ a_1:=a_1-2; \\ R_2(S_2);\ \text{if } S_2=255 \text{ then } a_2:=a_2-2; \\ \text{else } a_2:=a_2-2,\ Q_{stop}; \\ \text{end if}; \\ W_3(0),\ a_3:=a_3-2. \end{cases}$$

Устанавливаем головки в положение, когда они обозревают ячейку с начальным адресом:

$$Q_{15},\ \ 0_r, \rightarrow \forall r, r = \overline{0,3}\ \ a_r; = a_g, Q_{16}. \qquad .$$

Меняем позиции трех головок:

$$Q_{16},\ \ 0_r, \rightarrow \{a_1 := a_1 + 1, a_2 := a_2 + 2, a_3 := a_3 + 3\}, Q_{17}.$$

Считываем данные из ячеек памяти с увеличением кода адреса и сравниваем считанные данные с эталоном:

$$Q_{17}, 0_r, \rightarrow \left\{ \begin{array}{l} \text{if } a_3 \neq a_g \text{ then } F_{17}(Q_{17}, 0), Q_{17} \\ \text{else } F_{17}(Q_{17}, 0),\ Q_{19} \\ \text{end if} \end{array} \right\}.$$

Согласно функции $F_{17}(Q_{17}, 0)$ выполняются следующие действия:

$$F_{17}(Q_{17}, 0) = \forall r, r = \overline{0,3} \left\{ \begin{array}{l} R_r(S_r); \\ \text{if } S_r = 0 \text{ then } a_r := a_r + 4; \\ \text{else } a_r := a_r + 4, Q_{stop}; \\ \text{end if}; \end{array} \right.$$

В конце программы содержатся команды выдачи результатов тестирования в виде соответствующих сообщений: $Q_{19}, 0_r \rightarrow !$, «Ok» или $Q_{stop}, X_r \rightarrow !$ «Error».

Изложенные синтаксические особенности языка системы позволяют выделить элементарные операции, необходимые для проектирования алгоритмов и программ тестов.

Особенности применения системы визуализации алгоритмов тестов. При разработке алгоритмов диагностирования современных микросхем полупроводниковой памяти большое значение имеют визуализаторы алгоритмов, позволяющие в наглядной форме динамически отображать детали их работы [5, 6]. Поскольку система BAT имеет всего 256 ячеек, их состояние можно одновременно выводить на экран монитора и динамически демонстрировать выполняемые операции. Главное меню системы BAT приведено на рис. 2.

Для просмотра последовательности выполняемых операций реализован пошаговый режим работы системы, при котором на экран монитора осуществляется вывод строки команды и микроопераций, выполняемых в текущий момент времени.

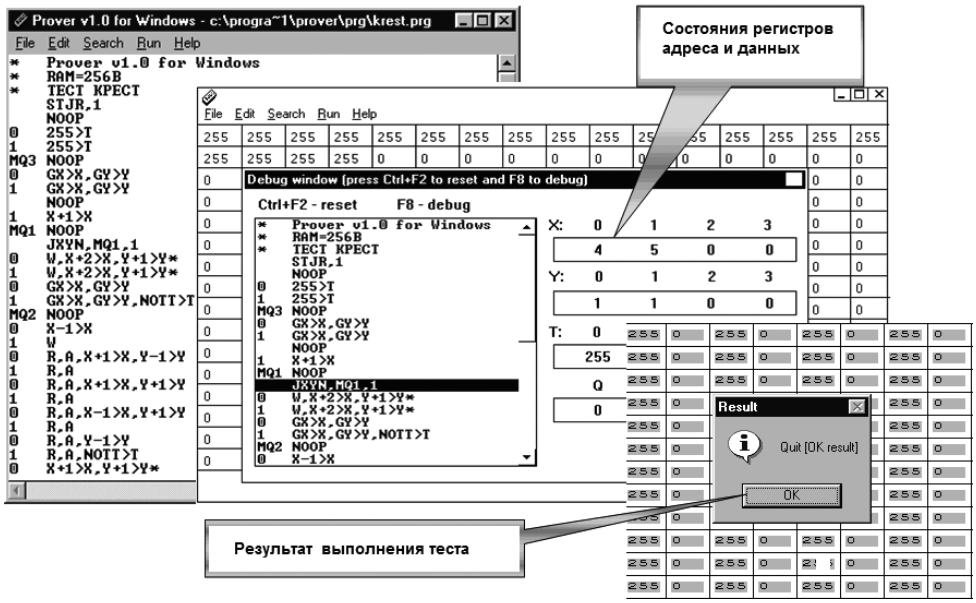

Рис. 2. Главное меню и результаты работы системы BAT

В автоматическом режиме при выполнении команд записи и считывания на экран выдается информация об адресе ячейки, с которой выполняется операция и символ, хранящийся в данной ячейке. Для удобства восприятия информации пользователь может изменить скорость заполнения ячейки при записи или считывании. При достижении команды анализа результатов диагностирования на дисплей выдается одно из сообщений о пригодности теста: «годен» или «брак».

Применяя систему BAT, можно использовать общую методологию построения алгоритмов и программ тестов, которая включает решение следующих задач:

выбор коэффициента распараллеливания операций формирования тестовых последовательностей в соответствии с заданным быстродействием микросхем памяти, что, в конечном счете, определяет число головок системы;

анализ алгоритма, представленного в операторном виде, и определение кратности числа операций в фрагментах алгоритма числу головок системы;

выполнение реконфигурации алгоритма, обеспечивающей кратность числа операций обращения к памяти в циклически повторяемых фрагментах алгоритма числу головок системы;

определение числа циклов повторения отдельных фрагментов алгоритма и микрооперации изменения кодов адреса и данных для каждого цикла;

декомпозиция исходной задачи на задачу меньшей размерности, т.е. проектирование и отлаживание алгоритма с помощью интерпретирующей системы ВАТ для микросхемы памяти базового объема емкостью 256 бит;

установление функциональной зависимости программных переменных от емкости тестируемой микросхемы;

формирование алгоритма для нового типа запоминающего устройства, имеющего, заданную емкость, заменой переменных, функционально зависимых от емкости памяти.

Поскольку для каждого нового типа устройства не требуется повторять весь процесс проектирования, уменьшаются затраты на разработку программ. Кроме того, трудоемкость локализации и устранения ошибок в программах, предназначенных для микросхемы памяти емкостью 256 байт, значительно ниже, чем трудоемкость отладки программ для тестирования запоминающих устройств большой емкости.

Выводы. Приведенный формат команд системы ВАТ обеспечивает синхронное управление перемещением нескольких головок, что обеспечивает увеличение быстродействия физически реализованного устройства диагностирования при распараллеливании процесса формирования тестовых воздействий. Разделение вычислительного процесса на элементарные действия позволяет определить набор микроопераций для параллельно выполняемых процессов.

В результате выполненных исследований получено теоретическое и практическое обоснование утверждения о том, что алгоритмы всех наиболее распространенных тестов диагностирования запоминающих устройств могут быть разработаны с помощью системы ВАТ и практически реализованы устройством диагностирования, имеющим мультипроцессорную структуру.

For the synthesis of algorithms and software diagnostic tests storage the system for the visualization algorithms tests is proposed containing the control unit, a square matrix of memory cells and four heads write (read). The example of synthesis using this system test march C is given. Reducing the complexity of design work is provided by dynamic imaging sequence performed diagnostic operations.

1. *Аль Мади М. К., Моамар Д. Н., Рябцев В. Г.* Алгоритмы тестового диагностирования полупроводниковых запоминающих устройств. — К. : «Корнійчук», 2008. — 220 с.
2. *Фалевич Б. Я.* Теория алгоритмов: Учеб. пособие. — М. : Машиностроение, 2004. — 160 с.
3. *Успенский В. А*. Машина Поста. — М. : Наука, 1988. — 96 с.
4. *Рябцев В. Г.* Проектування мобільних програм діагностування сучасних мікросхем пам'яті. // Вісник ЧДТУ. — 2002. — № 2. — С. 25—29.

5. *Абу Аль-Наадж М. В., Мовчан Ю. В., Рябцев В. Г.* Программные средства визуализации процесса моделирования микросхем памяти. // Моделювання та інформаційні технології. Зб. наук. праць ІПМЕ ім. Г.Є. Пухова НАН України. — 2002. — Вип. 12. — С. 85—90.
6. *Кудлаенко В. М., Моамар Д. Н., Рябцев В. Г.* Применение эмулятора машины Тьюринга для приобретения навыков синтеза алгоритмов. // Вісник ЧДТУ. — 2006. — № 4 — С. 25—28.

Поступила 01.12.09
после доработки 04.02.10

РЯБЦЕВ Владимир Григорьевич, д-р техн. наук, профессор, зав. кафедрой математических и компьютерных дисциплин Черкасского филиала Европейского университета (Украина). В 1969 г. окончил Харьковский авиационный ин-т. Область научных исследований — диагностика микросхем памяти запоминающих устройств.

МОАМАР Диаа Надим, аспирант Черкасского государственного технологического университета, который окончил в 2006 году. Область научных исследований — математическое моделирование.