УДК 681.326.7

АНАЛИЗ ИТЕРАТИВНОГО АЛГОРИТМА НЕРАЗРУШАЮЩЕГО ТЕСТИРОВАНИЯ ОЗУ

А.А. ИВАНЮК, С.Б. МУСИН

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки,6, Минск, 220013, Беларусь

Поступила в редакцию 3 ноября 2008

На каждой итерации исследуемого алгоритма неразрушающего тестирования оперативных запоминающих устройств (ОЗУ) инвертируются ячейки с адресами, выделенными по маске, которая формируется исходя из номера итерации алгоритма. Для сжатия тестовых реакций используется метод адаптивного сигнатурного анализа. В качестве целевых неисправностей определены многократные константные неисправности ОЗУ. Показано, что обнаруживающая способность повышается с увеличением итераций алгоритма. Определена формула вычисления верхней границы обнаруживающей способности и предельного количества итераций.

Ключевые слова: ОЗУ, неразрушающее тестирование, адаптивный сигнатурный анализ.

Введение

Для своевременного выявления неисправностей и сбоев при эксплуатации устройств критических приложений, где требования надежности и времени функционирования чрезвычайно высоки, используются методы контроля и неразрушающего тестирования ОЗУ. Методы контроля основаны на применении достаточно разработанной к настоящему времени теории помехоустойчивых кодов. При их использовании уменьшается полезная емкость устройства вследствие необходимости хранения проверочных разрядов. Кроме того, специфика работы ОЗУ требует быстрого и параллельного кодирования и декодирования информации. Процедуры контроля оказывают негативное влияние на временные параметры операций чтения и записи данных, поэтому на практике используются схемы кодирования и декодирования простых кодов: проверка на четность и код Хемминга. Таким образом, корректируются однократные ошибки, обнаруживаются ошибки 1–2 и нечетной кратности.

Методы неразрушающего тестирования основаны на алгоритмах генерации тестовых воздействий (маршевые алгоритмы тестирования) и анализа реакций ОЗУ (сигнатурный анализ). Маршевые алгоритмы имеют линейную сложность и позволяют выявлять многообразные целевые неисправности ОЗУ. Сигнатурный анализ (СА) строится на базе циклического кода и, в свою очередь, позволяет избежать сравнения с эталоном большого количества тестовых реакций путем их сжатия с определенной степенью достоверности в компактную оценку — сигнатуру. Применение СА для оперативного контроля ОЗУ существенно снижает производительность, так как необходимо повторное вычисление сигнатуры при каждом изменении хранимых данных.

В последнее время в научной литературе активно обсуждается проблема обнаружения многократных ошибок ОЗУ [1–6]. Дело в том, что уменьшение размеров запоминающих элементов, увеличение их плотности упаковки на кристалле и уменьшение напряжения питания является движущей силой полупроводниковой промышленности, платой за которую является снижение надежности устройства [4]. Снижение уровня критического заряда вызывает высо-

кую вероятность появления нерегулярных ошибок ОЗУ. Приведенные в [4, 5] результаты исследований показали, что кратность таких ошибок может быть четной и они не могут быть обнаружены.

Эффективность контроля при наличии нерегулярных ошибок может быть обеспечена путем использования композиционных кодов, а также кодов Рида−Маллера [1, 2].

Исследования, проведенные в работах [6, 7], показали, что эффективность тестирования может быть улучшена за счет многократного применения маршевых тестов. Улучшение обнаруживающей способности достигается за счет изменения содержимого ОЗУ в процессе его функционирования по назначению. Следует отметить, что, как показывают экспериментальные данные, полученные в рамках работы [8], даже при существенной нагрузке на систему, имеются блоки памяти, которые не подвержены существенным изменениям.

В настоящее время наблюдается также тенденция объединения методов тестирования и контроля. Так, в работах [9, 10] предлагается совместно использовать неразрушающие маршевые тесты ОЗУ и встроенную схему помехоустойчивого кодирования.

В [11] для контроля и тестирования используется метод адаптивного сигнатурного анализа (АСА). Развитием этого подхода стала работа [12], в которой предлагается использовать метод АСА для обнаружения неисправностей, проявление которых обеспечивает периодическое инвертирование содержимого ОЗУ. Экспериментальным путем было установлено, что для увеличения покрывающей способности теста при многократном его применении на каждой итерации следует инвертировать не все ячейки памяти. Инвертируются те ячейки, в двоичном представлении адреса которых бит, соответствующий номеру итерации, установлен в единицу. В данной статье проводится детальный анализ этого алгоритма с применением методики, предложенной в [13].

Алгоритм

Шаг 1: ОЗУ функционирует по назначению. При осуществлении операции записи данных производится сравнение предыдущего (хранимого) значения ячейки ОЗУ с новым (записываемым) значением. В случае неравенства этих значений выполняется адаптация эталонной сигнатуры ОЗУ.

Шаг 2: При необходимости тестирования хранимых данных функционирование ОЗУ приостанавливается.

Шаг 3: Осуществляется расчет рабочей сигнатуры с инвертированием содержимого памяти в соответствии со значением маски. Изначально инвертируются все ячейки памяти. На следующих итерациях значение маски увеличивается на единицу. Таким образом, будут инвертироваться ячейки с адресом, в двоичном представлении которого установлены в единицу биты соответствующие двоичному представлению номера итерации.

Шаг 4: Рабочая сигнатура сравнивается с эталонной. Результат сравнения определяет наличие либо отсутствие неисправностей в ОЗУ.

Шаг 5: Если нет необходимости в прерывании тестирования, перейти к 3. В противном случае ОЗУ переводится в режим функционирования по назначению, переход к 1.

Восстановление исходных (не инвертированных) значений хранимых данных не производится, так как они могут быть адаптированы при чтении в соответствии со значением маски. Таким образом, выполнение алгоритма может быть продолжено с той итерации, на которой было произведено прерывание.

Формализация

В качестве основного принимаем сформулированный нами в статье [13] постулат, согласно которому адресное пространство ОЗУ представляет собой проверочную матрицу кода Хемминга. В отличие от классического представления, где хранимые в ОЗУ данные считаются сообщением, которое передается по каналу связи, в нашей модели хранимые данные рассматриваются в качестве кодового слова линейного кода.

Принимаем, что изначально содержимое ОЗУ является корректным кодовым словом. Запись информации в ОЗУ будем рассматривать как санкционированное изменение первона-

чального кодового слова, поэтому данная операция сопровождается соответствующим изменением сигнатуры (синдрома кодового слова линейного кода). Для вычисления сигнатуры достаточно использовать адрес ячейки памяти, так как он эквивалентен соответствующему столбцу проверочной матрицы кода Хемминга (за исключением нулевого адреса) [14]. Вследствие необходимости учета данных и по нулевому адресу будем использовать проверочную матрицу расширенного кода Хемминга.

Представим набор данных, хранимых ОЗУ, как двоичный вектор-столбец **v**. Сжатие вектора **v** осуществляется с использованием проверочной матрицы **H**. Сигнатурой **s** вектора **v** будем называть двоичный вектор-столбец, полученный следующим образом: **s**=**Hv**.

Пусть ОЗУ **v** содержит 2^m ячеек памяти (m — разрядность шины адреса), Тогда $v_a \in \mathbf{v}$ — значение, хранящееся в ячейке по адресу **a**. Адаптация эталонной сигнатуры $\mathbf{s}_{ref}$ является последовательным вычислением произведения **Hv**, причем вместо столбцов матрицы расширенного кода Хемминга используется двоичное представление адреса, дополненное ведущей "1": $\mathbf{s}_{ref} \leftarrow \mathbf{s}_{ref} \oplus (1 \quad \mathbf{a})$.

При тестировании аналогичным образом вычисляется рабочая сигнатура $\mathbf{s}_{test}$, но предварительно к данным применяется тест — инвертирование в соответствии со значением маски *mask*.

На основании изложенного формализуем, приведенный в предыдущем разделе, алгоритм неразрушающего тестирования ОЗУ.

Инициализация: $mask \leftarrow 0$, $\mathbf{s}_{ref} \leftarrow \mathbf{0}$.

Шаг 1: **если** $v'_{\mathbf{a}} \cdot v''_{\mathbf{a}}$, **то** $\mathbf{s}_{ref} \leftarrow \mathbf{s}_{ref} \oplus (1 \quad \mathbf{a})$.

Шаг 2: $n \leftarrow 0$, $\mathbf{s}_{test} \leftarrow \mathbf{0}$.

Шаг 3: **пока** $n < 2^m$ **повторять**

если $(n \,\&\, mask) \oplus mask = 0$, **то** $v_n \leftarrow \overline{v}_n$, $\mathbf{s}_{test} \leftarrow \mathbf{s}_{test} \oplus (1 \quad \mathbf{a})$.

$n \leftarrow n+1$.

Шаг 4: **если** $\mathbf{s}_{ref} \neq \mathbf{s}_{test}$, **то** *обнаружена неисправность*.

Шаг 5: **если не** продолжать тестирование, **то** переход на *шаг 1*.

$mask \leftarrow mask + 1$.

если $mask = 2^m$, **то** $mask \leftarrow 0$.

переход на *шаг 3*.

Пусть вектор столбец $\mathbf{u}_{mask}$ содержит единичные значения для компонент, координат соответствующих маске *mask*, и нулевые для остальных компонент. Тогда, можно записать, что $\mathbf{s}_{test} = \mathbf{H}(\mathbf{v} + \mathbf{u}_{mask})$. На первой итерации выполнения алгоритма инвертируются все ячейки памяти. Но далее только те ячейки, адреса которых соответствуют номеру итерации алгоритма: инвертируются ячейки, младший бит адреса которых установлен в единицу ($1_{10}=1_2$), далее второй бит адреса ($2_{10}=10_2$), затем оба бита ($3_{10}=11_2$) и т.д. Так как маски последовательно выбираются из столбцов матрицы **H** (последовательность двоичных чисел), вектор $\mathbf{u}_{mask}$ является кодовым словом кода Хемминга и, следовательно, $\mathbf{Hu}_{mask}=0$. Поэтому, если ОЗУ не содержит неисправностей, изменение данных в соответствии с $\mathbf{u}_{mask}$ не оказывает влияние на сигнатуру, $\mathbf{s}_{test} = \mathbf{H}\ \mathbf{v} + \mathbf{u}_{mask} = \mathbf{Hv} + \mathbf{Hu}_{mask} = \mathbf{Hv} + \mathbf{0}$. В противном случае $\mathbf{u}_{mask}$ позволяет проявить неисправности. Для увеличения кратности проявляемых неисправностей значение маски изменяется от итерации к итерации, $\mathbf{s}_{test0}=\mathbf{H}(\mathbf{v}+\mathbf{u}_0)$, $\mathbf{s}_{test1}=\mathbf{H}(\mathbf{v}+\mathbf{u}_1)$, $\mathbf{s}_{test2}=\mathbf{H}(\mathbf{v}+\mathbf{u}_2)$, $\mathbf{s}_{test3}=\mathbf{H}(\mathbf{v}+\mathbf{u}_3)$ и т.д. Объединив полученные на каждой итерации сигнатуры в один вектор $\mathbf{s}_{tests}$, получим

$$\mathbf{s}_{tests} = \mathbf{s}_{test0}\mathbf{s}_{test1}\mathbf{s}_{test2}\mathbf{s}_{test3}\ldots = \left[\mathbf{H}\ \mathbf{v}+\mathbf{u}_0 \quad \mathbf{H}\ \mathbf{v}+\mathbf{u}_1 \quad \mathbf{H}\ \mathbf{v}+\mathbf{u}_2 \quad \mathbf{H}\ \mathbf{v}+\mathbf{u}_3 \quad \ldots\right] =$$

$$\mathbf{Hv}+\mathbf{Hu}_0 \quad \mathbf{Hv}+\mathbf{Hu}_1 \quad \mathbf{Hv}+\mathbf{Hu}_2 \quad \mathbf{Hv}+\mathbf{Hu}_3 \quad \ldots =$$

$$\mathbf{H}\ \mathbf{v}\ \ \mathbf{v}\ \ \mathbf{v}\ \ \mathbf{v}\ \ \ldots + \mathbf{u}_0\ \ \mathbf{u}_1\ \ \mathbf{u}_2\ \ \mathbf{u}_3\ \ \ldots\ .$$

Введем обозначение $\mathbf{U}=[\mathbf{u}_0\ \mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3\ \ldots]$. Матрица тестовых воздействий **U** позволяет оценить кратность целевых неисправностей, поэтому в следующем разделе проводится ее детальный анализ.

Анализ

Пусть **e** — вектор неисправностей ОЗУ. Если ОЗУ содержит ε неисправностей, то $w(\mathbf{e})=\varepsilon$, где $w=(\mathbf{e})$ — вес вектора **e**. Определим критерий маскирования неисправностей **e** как $\mathbf{s}_{tests}-\mathbf{s}_{refs}=0$, при $w(\mathbf{e})\neq 0$. Тогда

$$\mathbf{H}\ \mathbf{v}\ \mathbf{v}\ \mathbf{v}\ \mathbf{v}\ \ldots+\mathbf{e}\ \mathbf{e}\ \mathbf{e}\ \mathbf{e}\ \ldots+\mathbf{U}-\mathbf{H}\ \mathbf{v}\ \mathbf{v}\ \mathbf{v}\ \mathbf{v}\ \ldots=0\ \Rightarrow$$

$$\mathbf{H}\ \mathbf{e}\ \mathbf{e}\ \mathbf{e}\ \mathbf{e}\ \ldots+\mathbf{U}=0\ \Rightarrow\mathbf{H}\ \mathbf{e}\ \mathbf{e}\ \mathbf{e}\ \mathbf{e}\ \ldots=\mathbf{HU}.$$

Следовательно, неисправности [**e e e e** …] будут проявлены в результате тестовых воздействий **U** при условии линейной независимости ε столбцов матрицы **U**, т.е. минимальное расстояние линейного кода с проверочной матрицей **U** превышает количество неисправностей.

Рассмотрим гипотетическое ОЗУ, которое содержит 8 ячеек памяти, m=3. Тогда матрица тестовых воздействий будет иметь следующий вид:

$$\mathbf{U}=\begin{matrix}\mathbf{u}_0 & \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 & \mathbf{u}_5 & \mathbf{u}_6 & \mathbf{u}_7\end{matrix}=\begin{bmatrix}1&0&0&0&0&0&0&0\\1&1&0&0&0&0&0&0\\1&0&1&0&0&0&0&0\\1&1&1&1&0&0&0&0\\1&0&0&0&1&0&0&0\\1&1&0&0&1&1&0&0\\1&0&1&0&1&0&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}.$$

Заметим, что $\mathbf{u}_3=\mathbf{u}_1\cdot\mathbf{u}_2,\quad \mathbf{u}_5=\mathbf{u}_1\cdot\mathbf{u}_4,\quad \mathbf{u}_6=\mathbf{u}_2\cdot\mathbf{u}_4,\quad \mathbf{u}_7=\mathbf{u}_1\cdot\mathbf{u}_2\cdot\mathbf{u}_4$.

Перегруппировав столбцы, получили, что транспонированная матрица тестовых воздействий $\mathbf{U}^T$ совпадает с порождающей матрицей кода Рида–Маллера третьего порядка, r=3 $\mathbf{U}^T\equiv\mathbf{G}_{RM(3,3)}$ [14].

Таким образом, матрица **U** является проверочной матрицей кода дуального коду Рида–Маллера (3,3). Минимальное расстояние такого кода определяется по формуле

$$d^*=2^{m-(m-r-1)}=2^{r+1}=16.$$

Как видим, в проведении всех итераций алгоритма тестирования нет необходимости. Так как имеется всего 8 ячеек, а обнаруживаются неисправности до 16 кратности, для полного тестирования достаточно проведения лишь 4 шагов.

В общем случае при увеличении количества итераций теста происходит добавление новых столбцов в матрицу тестирования **U**. Общее количество шагов теста определяется количеством строк матрицы **U** и вычисляется следующим образом:

$$k=1+\sum_{i=1}^{r}C_m^i.$$

С ростом количества итераций растет порядок дуального кода Рида–Маллера r и соответственно количество обнаруживаемых неисправностей. Верхняя граница диапазона изменения r равна m–1, она определяется исходя из величины C_m^{m-1}. При этом обнаруживаются все неисправности до кратности $2^{m-1+1}=2^m$.

В случае максимального количества итераций m–1, обнаруживаются все неисправности до кратности 2^m.

Например, для обнаружения неисправностей до 16 кратности в 16 ячейках памяти достаточно 7 итераций теста, при этом матрица тестовых воздействий будет иметь следующий вид:

$$\mathbf{U}^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Заключение

Проведенный в статье анализ и математическая формализация итеративного алгоритма неразрушающего тестирования ОЗУ показал, что количество обнаруживаемых неисправностей растет с увеличением количества итераций алгоритма. Причем предельной величиной количества итераций будет значение m–1, где 2^m — количество ячеек ОЗУ. Для вычисления количества обнаруживаемых неисправностей следует использовать формулу определения минимального расстояния дуального кода Рида–Маллера.

Для дальнейших исследований алгоритма определены следующие направления: типы и адреса и взаимное расположение неисправностей с учетом отображения физического адреса ячейки на адрес блока, строки и столбца, а также характер изменений содержимого ОЗУ при функционировании по назначению.

ESTIMATION OF ITERATIVE ALGORITHM FOR TRANSPARENT RAM TESTING

A.A. IVANIUK, S.B. MUSIN

Литература

1. *Argyrides C., Himona S. L., Pradhan D.K.* // Proc. of Computing Frontiers. Italy, 2008. P. 353–358.
2. *Argyrides C., Zarandi H.R., Pradhan D.K.* // IEEE Int. Sym. on Circuits and Systems. USA, 2007. P. 356–358.
3. *Alzahrani F., Chen T.* // Proc. of the 1994 IEEE Int. Conf. on Computer Design: VLSI in Computer & Processors. 1994. P. 132–137.
4. *Dennard B.* // IEEE Design & Test of Computers. 2008. Vol. 25. P.188–191.
5. *Li X., Huang M.C., Shen K., Chu L.* // Proc. of the 3rd Workshop on Hot Topics in System Dependability. 2007. P. 13–19.
6. *Mrozek I., Yarmolik V.N.* // Proc. 6th Int. Conf. Computer Information Systems and Industrial Management Applications. Poland, 2003. P. 180–187.
7. *Yarmolik S.V., Kurbatskii A.N., Yarmolik V.N.* // Automatic Control and Computer Sciences. 2008. Vol. 42. P. 120–125.
8. *Solomon J.M., Huebner E., Bem D., Szezynska M.* // The Int. Journal of Digital Forensics & Incident Response. 2007. Vol. 4. P. 68–72.
9. *Thaller K., Steininger A.* // IEEE Transactions on Reliability. 2003. Vol 52, P. 413–422.
10. *Li J.-F.* Transparent-test methodologies for random access memories with/without ECC.
11. *Yarmolik V.N., Ivaniuk A.A., Krips M.* 5th Int. Workshop IEEE DDECS. Czech Rep., 2002. P. 360–365.
12. *Yarmolik V.N., Ivaniuk A.A., Petronenko D.S.* // Proc. Of Conf. on Mixed Design of Integrated Circuits and Systems. Poland, 2003. P. 360–363.
13. *Иванюк А.А., Мусин С.Б., Ярмолик В.Н.* // Микроэлектроника. 2007. № 3. С. 246–253.
14. *Блэйхут, Р.* Теория и практика кодов, контролирующих ошибки. М., 1986.