УДК 681.326:519.613

С.В. ЧУМАЧЕНКО, Е.И. ЛИТВИНОВА, А.В. ХАХАНОВА, В.А. ВАСИЛЕНКО

Харьковский национальный университет радиоэлектроники, Украина

ТЕХНОЛОГИЯ ВОССТАНОВЛЕНИЯ РАБОТОСПОСОБНОСТИ СИСТЕМЫ НА КРИСТАЛЛЕ

Предлагается технология минимального покрытия дефектных блоков резервными компонентами при восстановлении работоспособности логической части цифровой системы на кристалле. Рассматриваются общие положения и правила покрытия для матрицы конфигурируемых логических блоков с дефектными ячейками. Разрабатываются критерии покрытия дефектных ячеек. Приводятся примеры реализации алгоритма.

Ключевые слова: *верификация, восстановление работоспособности, дефект, система на кристалле.*

Введение

Миллиарды цифровых систем на кристаллах, используемых в мире, содержат до 16 типов разнообразных компонентов (процессор, память, логика, шины, специализированные вычислители), которые можно разделить на 2 подмножества: память (90%) и логика (10%). При этом дефекты, возникающие в памяти, ремонтируются встроенными средствами достаточно успешно всеми ведущими компаниями (Virage Logic, Intel). Но 10% логики практически не поддаются регулярным решениям в части встроенного ремонта. На сегодня в мире самая большая проблема на рынке электронных технологий – как восстанавливать работоспособность логической части цифровой системы на кристалле. Ввиду высокой рыночной привлекательности в работе рассматривается проблема диагностирования и ремонта памяти и логических ячеек, входящая в топ-десятку актуальных проблем компьютерной инженерии планеты от Gartner research group, путем переадресации неисправных ячеек на исправные компоненты из резервных строк, столбцов и ячеек. Стратегия работает на логических блоках, которые должны быть адресуемыми (и иметь запасные ремонтные блоки) или перепрограммируемыми на исправном пространстве кристалла для осуществления встроенного ремонта. Модели восстановления и ремонта модулей памяти SiP рассматривались в работах [1 – 6].

Следует также учесть, что уровень продаж компьютеров упал во 2 квартале 2009 года на 8% и составил 66 миллионов штук, но продажа ноутбуков при этом увеличилась на 20%. Что касается рынка чипов, то здесь фиксируется наивысший подъем продаж за последние 13 лет. Данное обстоятельство подтверждает закон Мура – транзистор сегодня ничего не стоит, платить пользователь будет за энергопотребление. Весь мир видит будущее в цифровых системах на кристаллах. Вывод – все рыночно-ориентированные идеи будут имплементированы в кристалл со специализированной функциональностью. В связи с этим актуальным представляется создание на кристалле инфраструктуры сервисного обслуживания, способной осуществлять встроенное диагностирование и ремонт, что существенно может повысить выход годной продукции и продлить время жизненного цикла цифрового изделия. Поэтому любое новое решение в данной области может быть интересным для рынка электронных технологий, что и определяет актуальность предложенной в исследовании технологии квазиоптимального покрытия дефектных блоков резервными компонентами.

Развитию теории и методов оптимизационного геометрического проектирования, в частности, изучению оптимизационной задачи размещения прямоугольных объектов посвящены работы [7 – 9].

В [7, 8] рассматриваются оптимизационные задача размещения прямоугольных объектов с переменными метрическими характеристиками в заданной области.

В [10] приведен анализ современных технологий встроенного сервисного обслуживания функциональностей цифровой системы в пакете. Рассмотрены особенности архитектуры «System-in-Package» и существующие стратегии восстановления работоспособности цифровых систем, а также метод оценки надежности восстановления их работоспособности.

В работе [11] рассматривается проблема адаптации технологий тестирования цифровых систем на кристаллах (System on Chip – SoC) для нового конструктивного поколения цифровых систем – System-in-Package (SiP), позволяющего эффективно и компактно имплементировать в кристаллы сверхслож-

ные специализированные вычислительные и радиочастотные устройства для рынка электронных технологий. Вместе с тем пакет кристаллов формирует спектр новых задач сервисного обслуживания SiP-функциональностей в реальном масштабе времени, которое существенно отличается от процессов встроенного диагностирования SoC. В связи с этим предлагается алгебрологический метод диагностирования и восстановления работоспособности функциональных логических блоков FPGA, основанный на использовании таблиц неисправностей и их анализе в реальном масштабе времени.

В [12] предлагается метод покрытия дефектных логических блоков цифровых систем на кристаллах ремонтными клетками путем обхода матрицы логических блоков в целях восстановления работоспособности компонентов программируемой логики. Метод позволяет получать решение в виде квазиоптимального покрытия всех дефектных блоков минимальным числом ремонтных клеток. Предлагается выбор одной из двух стратегий обхода строк или столбцов матрицы логических блоков на основании критериев структуризации, определяющих число неисправных блоков, приведенных к фактическому единичному каркасу модифицированной матрицы строк или столбцов.

Цель исследования – разработка технологии оптимального покрытия дефектных блоков резервными компонентами при восстановлении работоспособности логической части цифровой системы на кристалле.

Задачи исследования

1. Разработка общих положений и правил покрытия для матрицы конфигурируемых логических блоков с дефектными ячейками.

2. Разработка критериев покрытия дефектных ячеек.

3. Составление алгоритма, описывающего обход матрицы конфигурируемых логических блоков в целях построения покрытия.

4. Примеры реализации алгоритма.

1. Общие положения и правила покрытия

Рассматривается матрица конфигурируемых логических блоков с отмеченными дефектными ячейками. В соответствующей ей матрице смежности дефекту отвечает идентификатор 1. При выявлении дефектной ячейки выполняется ее покрытие. Покрывающий элемент представляет собой блок из 9 ячеек, которые образуют квадрат размера 3×3.

Способы покрытия дефектной ячейки. Покрытие дефектной ячейки, которая рассматривается как базовая, можно выполнить 9 способами (рис. 1).

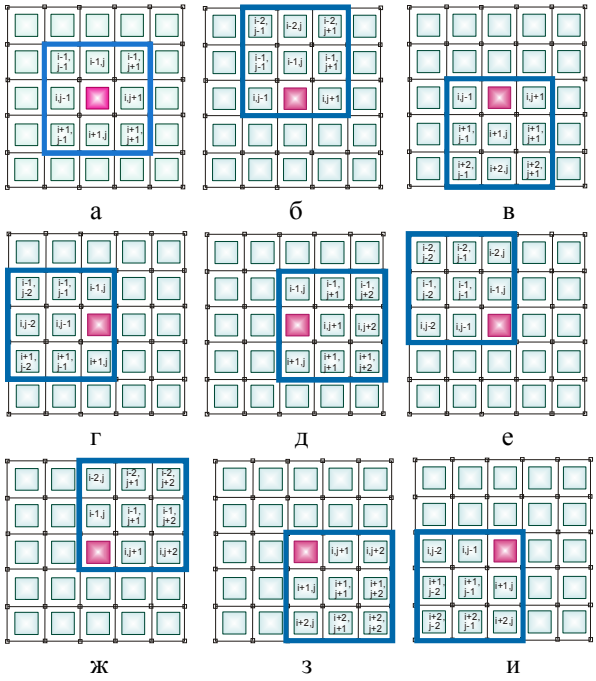

Рис. 1. Способы покрытия дефектной ячейки a_{ij}

На рис. 1 представлены следующие варианты покрытия:

$$а - a_{ij} \cup a_{i,j-1} \cup a_{i,j+1} \cup a_{i-1,j-1} \cup a_{i-1,j} \cup$$
$$\cup a_{i-1,j+1} \cup a_{i+1,j-1} \cup a_{i+1,j} \cup a_{i+1,j+1};$$

$$б - a_{ij} \cup a_{i,j-1} \cup a_{i,j+1} \cup a_{i-1,j-1} \cup a_{i-1,j} \cup$$
$$\cup a_{i-1,j+1} \cup a_{i-2,j-1} \cup a_{i-2,j} \cup a_{i-2,j+1};$$

$$в - a_{ij} \cup a_{i,j-1} \cup a_{i,j+1} \cup a_{i+1,j-1} \cup a_{i+1,j} \cup$$
$$\cup a_{i+1,j+1} \cup a_{i+2,j-1} \cup a_{i+2,j} \cup a_{i+2,j+1};$$

$$г - a_{ij} \cup a_{i,j-1} \cup a_{i,j-2} \cup a_{i-1,j} \cup a_{i-1,j-1} \cup$$
$$\cup a_{i-1,j-2} \cup a_{i+1,j} \cup a_{i+1,j-1} \cup a_{i+1,j-2};$$

$$д - a_{ij} \cup a_{i,j+1} \cup a_{i,j+2} \cup a_{i-1,j} \cup a_{i-1,j+1} \cup$$
$$\cup a_{i-1,j+2} \cup a_{i+1,j} \cup a_{i+1,j+1} \cup a_{i+1,j+2};$$

$$е - a_{ij} \cup a_{i,j-1} \cup a_{i,j-2} \cup a_{i-1,j} \cup a_{i-1,j-1} \cup$$
$$\cup a_{i-1,j-2} \cup a_{i-2,j} \cup a_{i-2,j-1} \cup a_{i-2,j-2};$$

$$ж - a_{ij} \cup a_{i,j+1} \cup a_{i,j+2} \cup a_{i-1,j} \cup a_{i-1,j+1} \cup$$
$$\cup a_{i-1,j+2} \cup a_{i-2,j} \cup a_{i-2,j+1} \cup a_{i-2,j+2};$$

$$з - a_{ij} \cup a_{i,j+1} \cup a_{i,j+2} \cup a_{i+1,j} \cup a_{i+1,j+1} \cup$$
$$\cup a_{i+1,j+2} \cup a_{i+2,j} \cup a_{i+2,j+1} \cup a_{i+2,j+2};$$

$$и - a_{ij} \cup a_{i,j-1} \cup a_{i,j-2} \cup a_{i+1,j} \cup a_{i+1,j-1} \cup$$
$$\cup a_{i+1,j-2} \cup a_{i+2,j} \cup a_{i+2,j-1} \cup a_{i+2,j-2}.$$

Ограничения покрытия дефектной ячейки. Если дефектная ячейка расположена в первых/последних двух строках/столбцах, очевидно, количество способов покрытия для нее ограничивается вариантами, представленными на рис. 2.

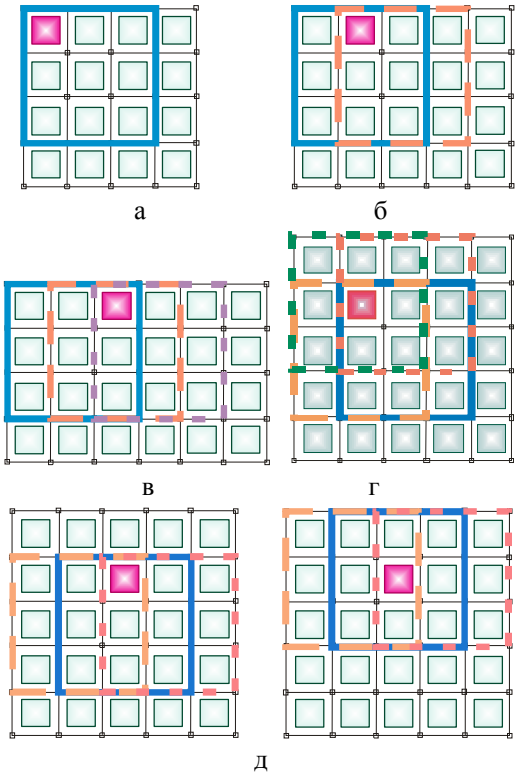

Рис. 2. Ограничения покрытия дефектной ячейки при расположении ее в двух крайних строках/столбцах

На рис. 2 представлены следующие варианты покрытия:

а – единственно возможный вариант построения покрытия при попадании дефекта в угловые ячейки $a_{11}, a_{1,N}, a_{N,1}, a_{N,N}$;

б – два способа покрытия при попадании дефекта в ячейки a_{12}, a_{21}, $a_{1,N-1}$, $a_{N-1,1}$, $a_{N-1,N}$, $a_{N,N-1}$, $a_{2,N}$, $a_{N,2}$;

в – три способа покрытия дефектной ячейки при расположении ее на позициях a_{13}, a_{31}, $a_{1,N-2}$, $a_{N-2,1}$, $a_{3,N}$, $a_{N,3}$, $a_{N,N-2}$, $a_{N-2,N}$;

г – четыре способа покрытия дефектной ячейки при расположении ее на позициях a_{22}, $a_{2,N-1}$, $a_{N-1,2}$, $a_{N-1,N-1}$;

д – шесть способов покрытия при попадании дефекта в ячейки a_{23}, a_{32}, $a_{2,N-2}$, $a_{N-2,2}$, $a_{3,N-1}$, $a_{N-1,3}$, $a_{N-1,N-2}$, $a_{N-2,N-1}$.

2. Критерии выбора и построения покрытия

При выборе покрывающего элемента предпочтение отдается квадрату с наибольшим весом. Вес определяется числом дефектных ячеек, которые попадают в покрывающий квадрат.

Например, на рис. 3 показаны варианты покрытия дефектной ячейки a_{ij}, когда возможен выбор покрывающего элемента с максимальным весом.

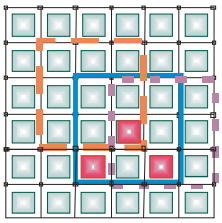

Рис. 3. Выбор покрывающего элемента – квадрата с наибольшим весом (сплошная линия)

Построчный просмотр матрицы. При реализации алгоритма построения покрытия выполняется просмотр матрицы по строкам, начиная с первой, в порядке слева направо. Затем осуществляется переход на вторую строку и просмотр элементов матрицы в обратном порядке. Таким образом, строки матрицы с нечетными номерами всегда просматриваются слева направо, после чего выполняется переход на нижнюю строку с четным номером и предпринимается продвижение в обратном порядке – справа налево.

Такой просмотр называется галсовым.

Для дефектной ячейки, встречающейся в строке, подбирается покрывающий квадрат с максимальным весом. После этого продолжается просмотр строки в целях выбора покрывающих квадратов для остальных дефектных ячеек данной строки.

При просмотре следующей строки часть дефектных ячеек в ней оказывается уже покрытой, тогда следует подобрать покрывающие квадраты для дефектных ячеек, оставшихся непокрытыми. При этом построенные ранее покрытия ограничивают выбор покрывающих квадратов для остальных дефектных ячеек, т.е. уменьшают в каждом случае их количество, что сокращает время поиска.

В случае, когда существует несколько способов покрытия дефектных ячеек квадратами с одинаковым максимальным весом, выбирается любой из них.

Если на каком-то этапе возникла ситуация, когда невозможно покрыть дефектную ячейку, следует вернуться назад к предыдущей дефектной ячейке в целях изменения ее покрытия квадратом, возможно, меньшего веса. При выборе нового покрытия предпочтение отдается квадрату с меньшим весом, покрывающему две соседние ячейки (расположенные по диагонали либо рядом в строке/столбце) и прилегающему к левому/правому краю (при продвижении слева направо или справа налево соответственно) с целью оставить большее количество свободных ячеек справа/слева соответственно для других покры-

тий. Другими словами, покрывающий квадрат смещается в сторону уже просмотренных к этому моменту строк/столбцов.

Просмотр матрицы по столбцам. Наравне с просмотром по строкам можно использовать продвижение по матрице сверху вниз по столбцам с последующим переходом вправо к соседнему столбцу и затем вверх.

Тогда столбцы с нечетными номерами будут просматриваться сверху вниз, а с четными – слева направо. При этом покрытие выбирается аналогично предыдущему также с учетом смещения покрывающего квадрата при возможности в сторону уже просмотренных ячеек.

Пример 1. Рассмотрим матрицу конфигурируемых логических блоков с отмеченными дефектными ячейками, представленную на рис. 4.

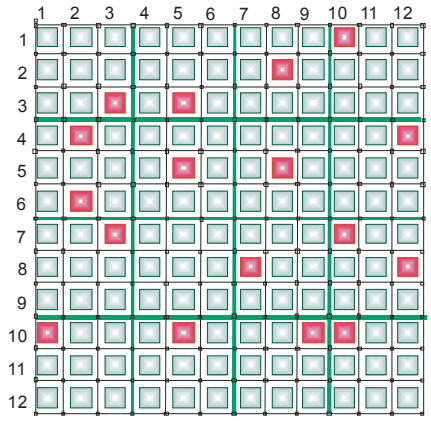

Рис. 4. Матрица логических блоков с дефектными ячейками

Согласно алгоритму, выполняется просмотр матрицы построчно. Для каждого дефектного элемента определяется максимальный вес его окрестности и выполняется группировка ячеек по блокам, т.е. объединение в покрытие, с учетом приведенных выше правил.

Просмотр строки с номером i=1 показывает, что оценка веса окрестности дефектного элемента $a_{1,10}$ равна $w(a_{1,10}) = 2$.

Тогда следует объединить дефектные ячейки $a_{1,10}$ и a_{28} в один покрывающий блок, а именно:

$$a_{18} \cup a_{19} \cup a_{1,10} \cup a_{28} \cup a_{29} \cup a_{2,10} \cup$$
$$\cup a_{38} \cup a_{39} \cup a_{3,10}.$$

При просмотре строки i=3 видно, что для ячейки a_{33} могло быть выбрано покрытие квадратом

$$a_{33} \cup a_{34} \cup a_{35} \cup a_{43} \cup$$
$$\cup a_{44} \cup a_{45} \cup a_{53} \cup a_{54} \cup a_{55}$$

с максимальным весом $w(a_{33}) = 3$.

Однако после просмотра строки i=4 справа налево оказалось бы, что дефектная ячейка a_{42} не может быть покрыта ни одним квадратом (рис. 5). В этом случае следует вернуться к предыдущей ячейке, которая была покрыта.

Если отсутствуют варианты изменения ее покрытия, то следует вернуться к предыдущей ячейке и изменить ее покрытие.

На рис. 5 это ячейка a_{33}.

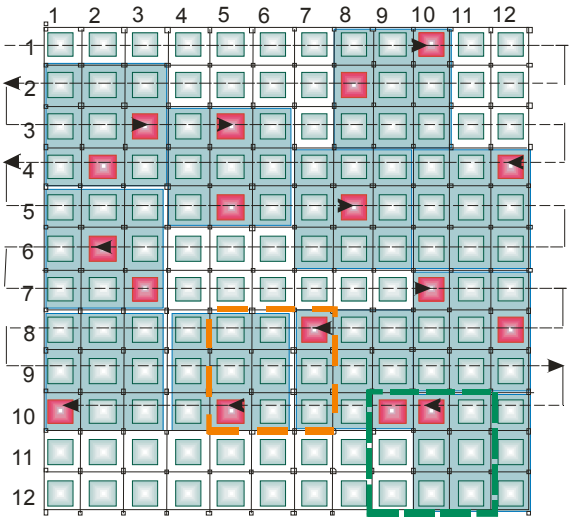

Рис. 5. Выбор покрытия при построчном продвижении

В результате получаем покрытие из 11 квадратов, среди которых 6 имеют вес 2 и 5 – вес 1. Из рис. 5, б видно, что расположенные рядом дефектные ячейки $a_{10,9}$ и $a_{10,10}$ можно было бы покрыть элементом:

$$a_{10,9} \cup a_{10,10} \cup a_{10,11} \cup a_{11,9} \cup a_{11,10} \cup$$
$$\cup a_{11,11} \cup a_{12,9} \cup a_{12,10} \cup a_{12,11}$$

или

$$a_{10,8} \cup a_{10,9} \cup a_{10,10} \cup a_{11,8} \cup a_{11,9} \cup$$
$$\cup a_{11,10} \cup a_{12,8} \cup a_{12,9} \cup a_{12,10}.$$

Тогда при изменении покрытия дефектной ячейки a_{87} на квадрат

$$a_{8,5} \cup a_{8,6} \cup a_{8,7} \cup a_{9,5} \cup$$
$$\cup a_{9,6} \cup a_{9,7} \cup a_{10,5} \cup a_{10,6} \cup a_{10,7}$$

получится покрытие из 10 квадратов, среди которых 7 имеют вес 2 и 3 – вес 1 (рис. 6).

Именно такое покрытие строится при просмотре матрицы по столбцам (рис. 6).

Таким образом, в примере 1 при просмотре матрицы по строкам получено квазиоптимальное покрытие, а по столбцам – оптимальное.

Как будет показано ниже, построчный просмотр матрицы практически всегда дает оптимальное покрытие.

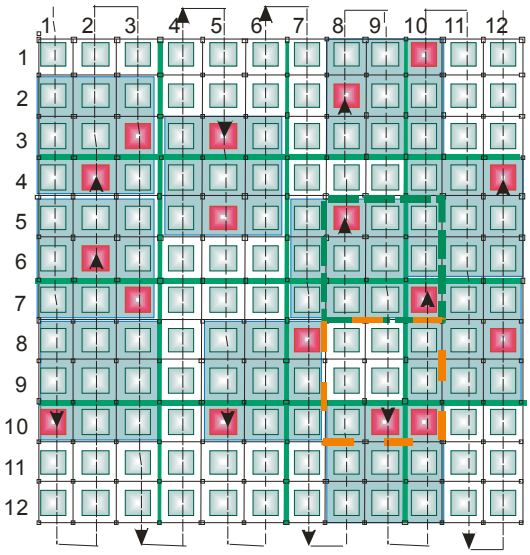

Рис. 6. Оптимальное покрытие матрицы конфигурируемых логических блоков

3. Описание алгоритма

1. Ввести матрицу $A = [a_{ij}]_1^n$.

2. Цикл по строкам (для i=1 до N)

2.1. Если номер строки нечетный i=2k+1, то перейти к п. 3 (цикл по столбцам)

2.2. Если номер строки четный i=2k, то перейти к п. 4 (цикл по столбцам – просмотр столбцов в обратном порядке от N до 1)

3. Цикл по столбцам (для j=1 до N)

4. Цикл по столбцам (для j=N до 1)

4.1. Вычислить веса соседних по отношению к базовой ячейке квадратов (обращение к ПОДПРОГРАММЕ 1/возврат из ПОДПРОГРАММЫ 1)

4.2. Выбрать в качестве покрывающего квадрат с максимальным весом

4.3. Исключить из рассмотрения ячейки, образующие покрывающий квадрат (обращение к ПОДПРОГАММЕ 2/ возврат из ПОДПРОГРАММЫ 2)

4.4. Возврат к п. 4 (продолжение просмотра по столбцам)

4.5. Возврат к п. 2 (переход к очередной строке)

5. Вывести результат - оптимальное (квазиоптимальное) покрытие в виде набора покрывающих квадратов, каждый их которых представляет собой объединение 9 ячеек.

ПОДПРОГРАММА 1 осуществляет оценку веса соседних по отношению к базовой дефектной ячейке квадратов.

ПОДПРОГРАММА 2 исключает из рассмотрения ячейки, образующие покрывающий квадрат, но хранит их в памяти в целях выполнения возвращения, если такое потребуется.

5. Примеры реализации алгоритма

Рассмотрим реализацию алгоритма на ряде примеров, где матрицы логических блоков с дефектными элементами сгенерированы случайным образом.

Пример 2. Рассмотрим матрицу конфигурируемых логических блоков с отмеченными дефектными ячейками, представленную на рис. 7.

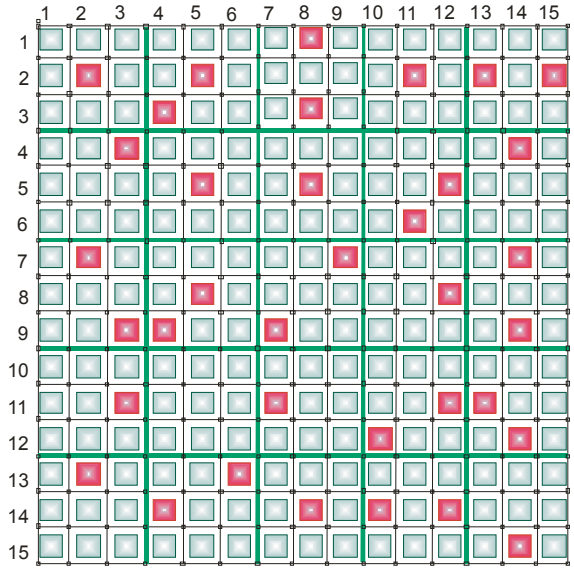

Рис. 7. Матрица логических блоков с дефектными ячейками для примера 2

В целях построения покрытия начинаем построчный просмотр матрицы согласно приведенным правилам, критериям и алгоритму. Маршрут прохождения матрицы при построении покрытия приведен на рис. 8. Стрелки на дефектных ячейках показывают, относительно какой базовой ячейки выполняется покрытие (обрамление покрывающими квадратами).

Представленное на рис. 8 оптимальное покрытие матрицы состоит из 17 блоков, из них: 3 имеют вес 1, 9 – с весом 2 и 5 – веса 3.

Пример 3. Рассмотрим варианты покрытия матрицы 15×15 конфигурируемых логических блоков с дефектными ячейками. Покрытие, полученное просмотром матрицы построчно, приведено на рис. 9. Оно состоит из 16 квадратов, из которых: 4 с весом 1, 7 – с весом 2, 2 – с весом 3, 3 – с весом 4.

На рис. 10 показан результат построения покрытия при просмотре матрицы по столбцам с продвижением слева направо.

Здесь также количество покрывающих элементов 16, при этом из них 4 с весом 1, 7 – с весом 2, 2 – с весом 3, 3 – с весом 4.

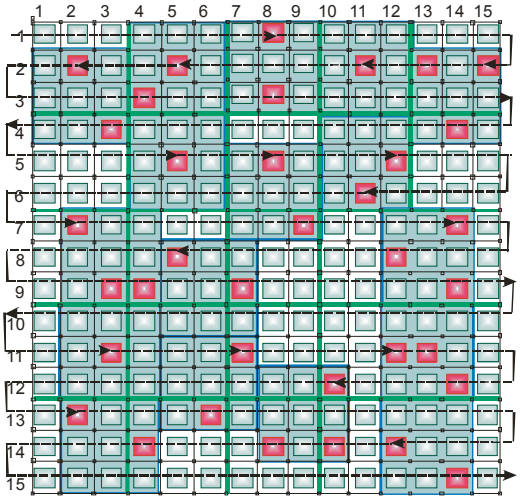

Рис. 8. Маршрут обхода матрицы при построении покрытия

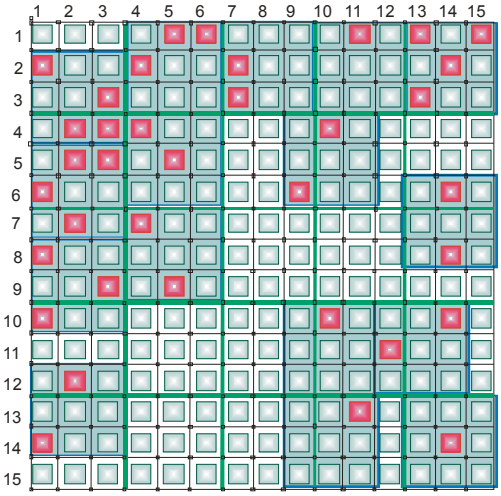

Рис. 9. Покрытие матрицы из примера 3

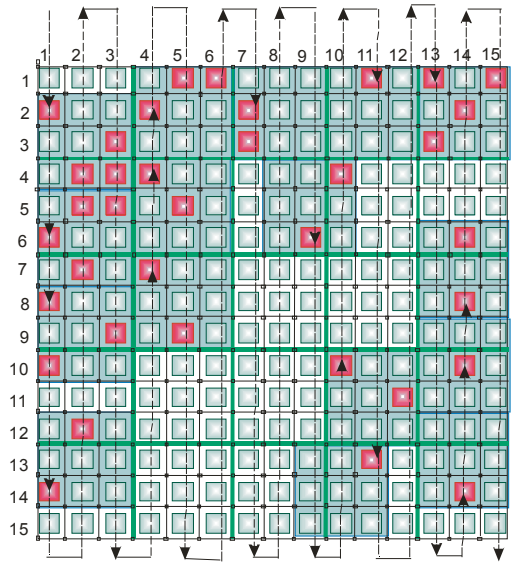

Рис. 10. Продвижение по столбцам при построении покрытия для матрицы из примера 3

Таким образом, два разных способа продвижения (по строкам и по столбцам) дают количественно одинаковый результат.

Пример 4. Рассмотрим построение покрытия для матрицы 15×15 конфигурируемых логических блоков с отмеченными дефектными ячейками. В целях построения покрытия предпримем обход матрицы по строкам (рис. 11).

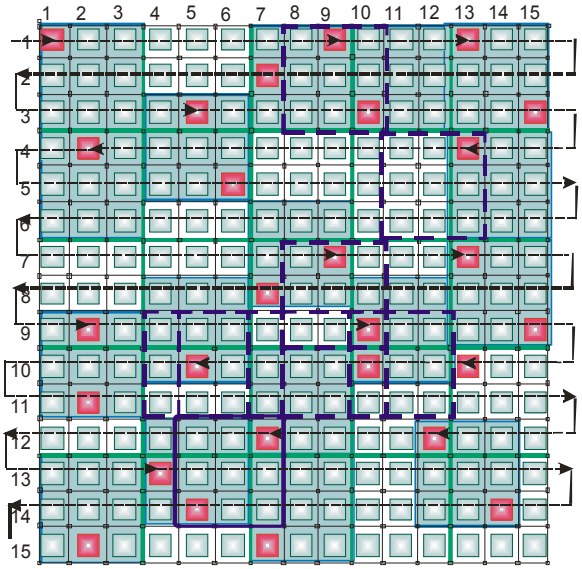

Рис. 11. Продвижение по строкам при построении покрытия для матрицы логических блоков с дефектными ячейками из примера 4

Пунктирными линиями на рис. 11 показаны покрытия, которые рассматривались при построении, но не были выбраны.

Так, для покрытия дефектной ячейки a_{19} при просмотре первой строки из двух блоков с весом 2 предпочтение отдается блоку $a_{17} \cup a_{18} \cup a_{19} \cup \cup a_{27} \cup a_{28} \cup a_{29} \cup a_{37} \cup a_{38} \cup a_{39}$, который объединяет часть уже просмотренных в первой строке ячеек (смещается влево по пути обхода строки), а не блоку

$$a_{18} \cup a_{19} \cup a_{1,10} \cup a_{28} \cup$$

$$\cup a_{29} \cup a_{2,10} \cup a_{38} \cup a_{39} \cup a_{3,10}.$$

При покрытии ячейки $a_{4,13}$ выбирается блок

$$a_{4,13} \cup a_{4,14} \cup a_{4,15} \cup a_{5,13} \cup a_{5,14} \cup a_{5,15} \cup$$

$$\cup a_{6,13} \cup a_{6,14} \cup a_{6,15}$$

с весом 1, а не

$$a_{4,11} \cup a_{4,12} \cup$$

$$\cup a_{4,13} \cup a_{5,11} \cup a_{5,12} \cup a_{5,13} \cup a_{6,11} \cup a_{6,12} \cup \cup a_{6,13}$$

с таким же весом, поскольку просмотр четвертой строки выполняется справа налево и первый блок включает уже просмотренные к этому моменту

ячейки, т.е. выбираемый блок сдвигается в сторону уже пройденных ячеек – вправо по пути обхода.

Таким образом, оптимальное покрытие матрицы, представленной на рис. 13, состоит из 17 элементов: 8 – веса 1, 9 – веса 2.

Заключение

В результате исследований разработан метод квазиоптимального покрытия неисправных адресуемых ячеек цифровых систем на кристаллах ремонтными клетками, что позволяет существенно повысить выход годной продукции, реализованной в чипах программируемой логики.

Сравнение с аналогами. Существующие аналоги, как правило, ориентированы на оптимизацию размещения компонентов в объемах или на плоскостях. Отличие предложенного метода заключается в оптимизации покрытия неисправных компонентов ремонтными клетками, причем неисправности могут быть соседними или отстоять далеко друг от друга. Поэтому здесь функция цели – минимизировать число ремонтных клеток, которое полностью покроет все дефектные адресуемые ячейки кристалла.

Научная новизна. Предложенный метод в своей постановочной части обладает определенной оригинальностью. Особенностью алгоритма квазиоптимального покрытия, реализованного в программном коде, является галсовый обход всех ячеек матрицы по строкам (столбцам) в целях определения и выбора оптимального решения задачи покрытия. Выбор покрывающих элементов с максимальным весом и построенные ранее покрытия ограничивают количество покрывающих квадратов для остальных дефектных ячеек, что сокращает время поиска.

Практическая значимость предложенного метода заключается в возможности его применения для встроенного ремонта компонентов цифровых систем на кристаллах, включающих адресуемые ячейки памяти или логические адресуемые блоки. В общем случае данный метод может быть применен для восстановления работоспособности любых адресуемых компонентов, представленных на плоскости.

Перспективы исследований. Имплементация технологии восстановления работоспособности цифровых систем на кристаллах для повышения надежности функционирования и выхода годной продукции в виде изделий, работающих в критических средах таких, как космос, авиация.

Литература

1. Memory Repair Primer – A guide to understanding embedded memory Repair options and issues. Logic Vision. 2007.

2. Youngs L. Mapping and Repairing Embedded-Memory Defects / L. Youngs, S. Paramanandam // IEEE Design and Test of Computers. – 1997. – P. 18–24.

3. Zorian Y. Embedded-Memory Test and Repair: Infrastructure IP for SoC Yield / Y. Zorian, S. Shoukourian // IEEE Design and Test of Computers. 2003. P. 58–66.

4. Huang R. Economic Aspects of Memory Built-in Self-Repair / R. Huang, Ch.Chen, Ch. Wu // IEEE Design & Test. – 2007. – P. 164–172.

5. Choi M. Optimal Spare Utilization in Repairable and Reliable Memory Cores / M. Choi, N. Park, F. Lombardi, Y. B. Kim, V. Piuri // 2003 Int. Workshop on Memory Technology, Design and Testing (MTDT'03). – 2003. – P. 64–71.

6. Ohler Ph. An Integrated Built-In Test and Repair Approach for Memories with 2D Redundancy / Ph. Ohler, S. Hellebrand, H.-J. Wunderlich // 12th IEEE European Test Symposium (ETS'07). – 2007. – P. 91-96.

7. Новожилова М.В. Метод решения задачи размещения прямоугольников с переменными метрическими характеристиками / М.В. Новожилова, И.А. Чуб, М.Н. Мурин // Радиоэлектроника и информатика. – 2008. – № 4. – С. 134-141.

8. Чуб И.А. Модификация точного метода решения задачи размещения прямоугольных объектов / И.А. Чуб, М.В. Новожилова // АСУ и ПА. – 2008. – Вып. 145. – С. 57-63.

9. Стоян Ю.Г. Математические модели и оптимизационные методы геометрического проектирования / Ю.Г. Стоян, С.В. Яковлев – К.: Наук. думка, 1986. – 266 с.

10. Литвинова Е.И. Технологии встроенного ремонта компонентов System-in-Package / Е.И. Литвинова // АСУ и приборы автоматики. – 2008. – Вып. 145.– С. 40-48.

11. Литвинова Е.И. Технологии диагностирования и восстановления System-in-Package / Е.И. Литвинова // АСУ и приборы автоматики. – Вып. 146. – 2009. – С. 4-21.

12. Литвинова Е.И. Метод покрытия неисправных логических блоков цифровых систем на кристаллах ремонтными клетками / Е.И. Литвинова // Радиоэлектроника и информатика. – 2009. – №1. – С. 45-51.

Поступила в редакцию 13.01.2010

Рецензент: д-р техн. наук, проф., проф. кафедры Ф.В. Новиков, Харьковский национальный экономический университет, Харьков.

ТЕХНОЛОГІЯ ВІДНОВЛЕННЯ ПРАЦЕЗДАТНОСТІ СИСТЕМИ НА КРИСТАЛІ

С.В. Чумаченко, Є.І. Литвинова, Г.В. Хаханова, В.О. Василенко

Запропоновано технологію мінімального покриття дефектних блоків резервними компонентами при відновленні працездатності логічної частини цифрової системи на кристалі. Розглянуто загальні положення та правила покриття для матриці конфігуруємих логічних блоків з дефектними комірками. Розроблено критерії покриття дефектних комірок. Наведено приклади реалізації алгоритму.

Ключові слова: верифікація, відновлення працездатності, дефект, система на кристалі.

TECHNOLOGY FOR REPAIRING OF SYSTEM-ON-CHIP

S.V. Chumachenko, E.I. Litvinova, A.V. Hahanova, V.A. Vasilenko

The technology for the minimum covering of faulty blocks by spares, when repairing of SoC logic part, is proposed. General provisions and rules for covering a matrix of configurable logic blocks with faulty cells are considered. Coverage criteria of faulty cells are developed. Examples of algorithm implementation are presented.

Key words: verification, repairing, fault, system-on-a-chip.

Чумаченко Светлана Викторовна, д-р техн. наук, профессор кафедры АПВТ Харьковского национального университета радиоэлектроники; e-mail: ri@kture.kharkov.ua.

Литвинова Евгения Ивановна – канд. техн. наук, доцент, доцент кафедры технологии и автоматизации производства РЭС и ЭВС Харьковского национального университета радиоэлектроники; e-mail: kiu@kture.kharkov.ua.

Хаханова Анна Владимировна – канд. техн. наук, ст. преподаватель кафедры АПВТ Харьковского национального университета радиоэлектроники; e-mail: hahanov@kture.kharkov.ua.

Василенко Василина Александровна – аспирантка кафедры АПВТ Харьковского национального университета радиоэлектроники; e-mail: kiu@kture.kharkov.ua.