УДК 658.512.011:681.326:519.713

Хаханов В. И.[1], Мурад Али Аббас[2], Литвинова Е. И.[3], Хаханова И. В.[4]

[1,3,4]Д-р техн. наук, профессор Харьковского национального университета радиоэлектроники
[2]Аспирант Харьковского национального университета радиоэлектроники

МОДЕЛИ ВСТРОЕННОГО РЕМОНТА ЛОГИЧЕСКИХ БЛОКОВ

Предложены модели комбинационных схем, ориентированные на решение практических задач встроенного восстановления работоспособности компонентов логических устройств. Логическая схема дополняется операционным и управляющим автоматами моделирования цифровых устройств, что увеличивает время обработки и аппаратные затраты для создания оболочки адресуемых элементов. Структуры также можно использовать для аппаратного моделирования функциональностей цифровых проектов на основе использования PLD, что дает возможность существенно повысить быстродействие верификации программных моделей. Предложенное решение задачи встроенного ремонта логических элементов комбинационных схем дает возможность комплексно решать проблему автономного восстановления работоспособности цифровых систем на кристаллах за счет временной и аппаратной избыточности проекта [1–23].

Ключевые слова: комбинационная схема, восстановление работоспособности, моделирование, верификация, программная модель.

ВВЕДЕНИЕ

Качество – совокупность свойств, которые характеризуют степень удовлетворения потребителя при использовании изделия в соответствии с назначением. Надежность изделия как одного из показателей качества формируется четырьмя факторами: безотказности, долговечности, ремонтопригодности и сохраняемости. Надежность – свойство объекта сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях применения, технического обслуживания, хранения и транспортирования [1]. Надежность в узком смысле есть свойство объекта сохранять работоспособное состояние в течение определенного времени (жизненного цикла) или некоторой наработки на отказ.

Первым вариантом повышения надежности системы в режиме online является использование кодов, исправляющих ошибки. Но они теряют способность к компенсации в случае возникновения постоянных во времени функциональных или структурных нарушений. Кроме того, увеличивая надежность системы, избыточные коды уменьшают время наработки на отказ, ввиду существенных накладных аппаратных затрат, связанных с исправлением ошибок в процессе функционирования изделия [19]. Другая стратегия, связанная с реконфигурированием функциональности при возникновении отказов внутри одного кристалла достаточно популярна, но имеет два недостатка: 1) существенное время выполнения данной операции, несовместимое с функционированием критических систем; 2) значительное усложнение процесса реконфигурирования, связанное с существованием неисправных областей кристалла. Третьим вариантом повышения надежности цифровой системы на кристалле может служить внесение избыточных элементов к основной функциональности, которые предназначены компенсировать негативные последствия неисправных компонентов путем переадресации в цикле тестирования и обнаружения неисправностей в изделии, который по времени существенно меньше цикла реконфигурирования функциональной области кристалла. Реализация третьего варианта, предложенного в [20–23], заключается в добавлении (модификации) трех компонентов (запасные функциональные элементы и порты считывания/записи данных, расширенные мультиплексоры для коммутации первых двух) к функциональности SoC, которая обеспечивает возможность замены отказавших элементов в цикле BIST/BISR.

Надежность некоторой дискретной области (компонента) цифровой системы характеризуется критерием с экспоненциальным распределением, зависящим от интенсивности λ некоррелированных отказов, постоянных во времени: $R_A(t) = e^{-\lambda t}$. Для рассмотрения надежности первоначальной структуры в целом можно использовать следующую оценку [20]: $R_O(t) = R_A(t)^{A_O}$, $A_O = M \times A_{FU}$. При этом система не имеет возможности компенсировать возникновение отказов, приводящих ее в неработоспособное состояние. В случае рассмотрения аппаратной сложности (Area) системы A_O с избыточностью, содержащей M основных компонентов и N запасных, для восстановления работоспособности при возникновении отказов в функциональных модулях, необходимо использовать следующие преобразования в определении оценки надежности R_{BISR}:

$$R_{BISR} = R_{SW} \times R_{MN};$$

$$R_{MN} = \sum_{i=M}^{N} \binom{N}{i} R_{FU}^{i} \times [1 - R_{FU}]^{N-i};$$

$$R_{SW} = R_O{}^{(A_{SW})/A_O} = R_O{}^{(A_{SW})/(M \times A_{FU})};$$

$$R_{FU} = R_O{}^{(A_{FU})/A_O} = R_O{}^{1/(M)} = R_O{}^{-M};$$

$$R_{BISR} = R_O{}^{(A_{SW})/(M \times A_{FU})} \times \sum_{i=M}^{N} \binom{N}{i} R_O^{i-M} \times [1 - R_O^{-M}]^{N-i}.$$

Здесь A_{FU}, A_{SW} – число дискретных элементов (аппаратная сложность) в функциональном и запасном компоненте структуры A_O; R_{MN}, R_{SW}, R_{FU} – показатели надежности системы с избыточными элементами, дополнительного оборудования для коммутации, функционального компонента соответственно. Лучший показатель надежности системы фиксируется, когда параметр коммутационного (адресного) оборудования A_{SW} стремится к нулю. Таким образом, надежность системы с избыточностью всегда будет выше, чем структура без запасных компонентов, если коммутационное оборудование будет меньше, чем функциональное. Исключение составляет случай, когда M=1, N=0, где надежности с избыточностью и без резервирования равны.

Что касается оценки жизненного цикла изделия (lifetime), то здесь увеличение числа запасных элементов, при одинаковом соотношении $(A_{SW})/A_O$, приводит к увеличению параметра (Mean Time To Failure – MTTF) – среднее время наработки на отказ, о чем свидетельствует следующая формула:

$$MTTF_{BISR} = \int_0^{\infty} R_{BISR}(t)dt.$$

Тем не менее, использовать значительное число избыточных компонентов не всегда практически целесообразно, поскольку аппаратная сложность управления при мультиплексировании запасных элементов существенно (линейно) зависит от их числа:

$$A_{SW} = (N_{sp} + 1) \times N_{IO} \times A_{switch}.$$

Здесь представлены параметры: 1) число избыточных элементов; 2) количество портов или входных и выходных переменных, переадресуемых в процессе вынужденной коммутации при возникновении отказа; 3) аппаратная сложность при реализации одного переключения.

Интерес представляет вычисление аппаратной избыточности при сегментации цифрового изделия на части, где каждый сегмент имеет собственные запасные элементы вместо увеличения общей избыточности для всей схемы:

$$A_{SW} = \sum_{i=1}^{N_{seg}} (N_{sp_i} + 1) \times N_{IO_i} \times A_{switch_i}.$$

Преимущества сегментации заключаются в получении более пологой кривой линейного возрастания аппаратных затрат на мультиплексирование запасных элементов, проводимого внутри сегмента. Недостаток в данном случае – строгое ограничение использования запасных компонентов сегмента внутри данной области.

Выводы: 1. Предложенная аналитическая модель вычисления надежности, времени наработки на отказ, а также стратегия сегментации позволяют без проведения процедур синтеза и имплементации оценить качество (надежность) проекта. 2. Метод BISR не ориентирован на замену отказавших элементов в режиме online. Тем не менее, он может быть использован для восстановления работоспособности цифрового изделия в режиме встроенного ремонта после обнаружения функционального нарушения существующими online стратегиями тестирования. 3. Решение проблемы адресуемости логических блоков есть путь к встроенному ремонту всех компонентов цифровых систем на кристаллах.

Понятие адресного выполнения логических операций, реализованных на элементах памяти LUT в программируемых логических устройствах (PLD), дает потенциальную возможность создавать на кристалле только адресное пространство для встроенного восстановления работоспособности всех компонентов, участвующих в формировании функциональности [1–18]. Актуальность создания адресного пространства для всех компонентов подтверждается следующим распределением логики и памяти на кристалле, представленным на рис. 1.

Тенденция к увеличению памяти влечет возможность встроенного восстановления работоспособности отказавших ячеек за счет выделенных дополнительных ресурсов для их ремонта (spare logic cells). Проблема автономного устранения дефектов (самовосстановления работоспособности) логических элементов связана с отсутствием у них адресов. Но решить проблему можно, если связи между элементами логики сделать гибкими с помощью программы описания структуры, помещенной в память, которая соединит логические компоненты в схему. Кроме структуры взаимодействия элементов память должна содержать порядок их обработки. В случае возникновения дефекта в одном из адресуемых логических элементов, система встроенного тестирования восстановит его работоспособность путем переадресации на заведомо исправный аналог из ремонтного запаса.

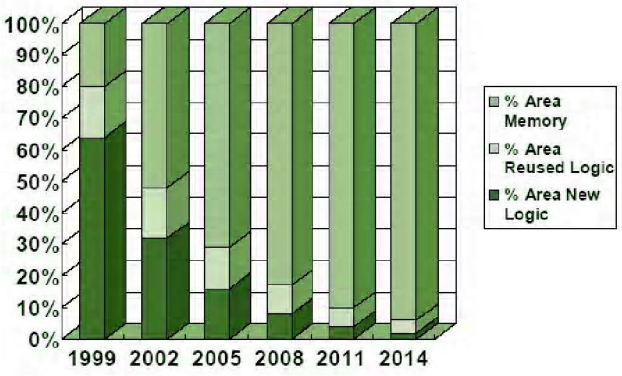

Рис. 1. Соотношение памяти и логики на кристалле

Цель – повышение качества и надежности цифровых систем на кристаллах путем создания инфраструктуры встроенного тестирования, диагностирования, оптимизации и восстановления работоспособности за счет аппаратной избыточности и уменьшения быстродействия выполнения функциональных операций [2–18].

Задачи и источники: 1. Анализ современных технологий встроенного восстановления работоспособности компонентов SoC [7–22]. 2. Разработка математической модели встроенного ремонта логических элементов, входящих в комбинационную структуру функциональности, имплементированную в SoC [7–19]. 3. Создание операционного и управляющего автоматов для эмулирования функциональности комбинационной схемы в кристалле PLD [2–6].

1. МОДЕЛЬ КОМБИНАЦИОННОЙ СТРУКТУРЫ

Немногочисленные работы, посвященные восстановлению работоспособности логических схем [9, 20–23], описывают две идеи. Первая заключается в реконфигурации структуры логических элементов в режиме off-line, которая обеспечивает возможность замены каждого из неисправных примитивов. Вторая создает условия для замены неисправных элементов путем использования запасных логических компонентов и расширения мультиплексоров для переадресации отказавших примитивов. Далее предлагается в качестве примера для рассмотрения теории и практики адресации примитивов комбинационных схем в целях встроенного ремонта функциональных нарушений логических элементов использовать описание простейшей схемной структуры (рис. 2).

Она содержит шесть однотипных логических элементов, которые можно представить в адресном пространстве следующим списком (двумерным массивом):

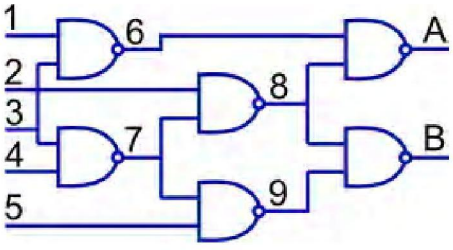

Рис. 2. Пример схемной структуры из неадресуемых элементов

$S =$

$No =$	1	2	3	4	5	6	7	8	9
$P =$	1	1	1	1	1	1	1	1	1
$L_1 =$	1	3	2	7	6	8	X	X	X
$L_2 =$	3	4	7	5	8	9	X	X	X
$L_3 =$	6	7	8	9	A	B	Y	Y	Y

$No =$	1	2	3	4	5	6	7	8	9	A	B
$M =$	1	1	1	1	1	0	0	1	1	1	1

$F(1) =$

X_1	X_2	Y
0	0	1
0	1	1
1	0	1
1	1	0

Каждый столбец соответствует логическому элементу схемы, а примитивы с номерами 7, 8, 9 являются запасными, которые используется для замены любых трех из шести элементов при диагностировании в последних каких-либо функциональных нарушений. В строке P указаны типы примитивов, ниже – номера входных и выходных переменных, вектор моделирования M содержит результат анализа входного слова 11111 на схемной структуре, представленной рис. 3.

Процесс-модель формирования выходных значений схемы в зависимости от конкатенированных состояний входов, формирующих адрес ячейки состояния выхода, можно представить в абстрактном или аппаратно ориентированном, на использование вектора состояния M, виде:

$$\begin{bmatrix} Y_6 = P_1(X_1 * X_3); \\ Y_7 = P_2(X_3 * X_4); \\ Y_8 = P_3(X_2 * X_7); \\ Y_9 = P_4(X_7 * X_5); \\ Y_A = P_5(X_6 * X_8); \\ Y_B = P_6(X_8 * X_9). \end{bmatrix} \rightarrow \begin{bmatrix} M_6 = P_1(M_1 * M_3); \\ M_7 = P_2(M_3 * M_4); \\ M_8 = P_3(M_2 * M_7); \\ M_9 = P_4(M_7 * M_5); \\ M_A = P_5(M_6 * M_8); \\ M_B = P_6(M_8 * M_9). \end{bmatrix}$$

Здесь M – вектор состояния линий схемы, P_i – логическая функция И-НЕ, имеющая два входа, реализованная в виде элемента памяти LUT. Поскольку все шесть примитивных элементов реализуют одну логическую функцию И-НЕ, то предыдущее выражение можно упростить путем замены всех структурных элементов одним функциональным примитивом F с последующим использованием для операции конкатенации двумерного массива линий связи (L) между входами и выходами логических элементов:

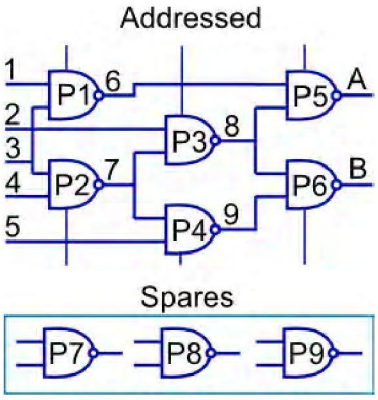

Рис. 3. Пример схемной структуры из адресуемых и запасных элементов

$$\begin{bmatrix} M_6 = F(M_1 * M_3); \\ M_7 = F(M_3 * M_4); \\ M_8 = F(M_2 * M_7); \\ M_9 = F(M_7 * M_5); \\ M_A = F(M_6 * M_8); \\ M_B = F(M_8 * M_9). \end{bmatrix} \rightarrow \begin{bmatrix} M_6 = F[M(L_{11}) * M(L_{12})]; \\ M_7 = F[M(L_{21}) * M(L_{22})]; \\ M_8 = F[M(L_{31}) * M(L_{32})]; \\ M_9 = F[M(L_{41}) * M(L_{42})]; \\ M_A = F[M(L_{51}) * M(L_{52})]; \\ M_B = F[M(L_{61}) * M(L_{62})]. \end{bmatrix}$$

Таким образом, можно синтезировать структуру для реализации процесс-модели схемы, имеющей двухвходовые функциональные примитивы, в следующем виде:

$$M(L_{is_p}) = F[M(L_{ij}) * M(L_{ir})] = F[M(L)].$$

Учитывая факт, что все вычисления в схеме привязаны к структурным элементам, которые имеют идентификатор логической операции, предыдущую формулу можно трансформировать к виду:

$$M(L_{is_p}) = P_i[M(L_{ij}) * M(L_{ir})] = P[M(L)].$$

В общем случае структура модели функциональности, ориентированной на реализацию в кристалле PLD, содержит пять компонентов:

$$S = < P, F, M, L, T >,$$

где $P = (P_1, P_2, ..., P_i, ..., P_n)$; $F = (F_1, F_2, ..., F_j, ..., F_m)$;

$M = (M_1, M_2, ..., M_r, ..., M_k)$; $L = [L_{pq}]; p = \overline{1, n}; \ q = \overline{1, s_p}$;

$T = [T_{te}]; t = \overline{1, \eta}; \ e = \overline{1, \mu}; \ M(L) = P[M(L)]$.

Здесь представлены: 1) примитивы схемной структуры P, определенные идентификаторами типа функциональности (номер или код команды); 2) типы функциональных элементов F – набор элементов памяти LUT, из которых реализуются примитивы, а также избыточные элементы для ремонта функциональностей; 3) вектор моделирования M (двоичный), определяющий состояния всех линий (входные, внутренние, выходные); 4) матрица эквипотенциальных линий связи L для объединения n логических элементов в структуру; 5) матрица входных тестовых (рабочих) наборов T. Обработка схемы (processing) схемы в кристалле сводится к определению адреса, составленного двоичными битами вектора мо-

делирования, по которому находится логическая функция. Каждый примитив имеет цикл обработки, содержащий три процедуры.

1. Адресное считывание номеров входных переменных из соответствующего столбца матрицы L для формирования адреса состояния входной переменной вектора моделирования: $A = L_{ij}$, $i = \overline{1,n}$; $j = \overline{1,s_p - 1}$.

2. Формирование адреса (двоичного кода) для вычисления логической функции путем конкатенации соответствующих состояний входных переменных в векторе моделирования $A = M(L_{ij}) * M(L_{ir})$.

3. Запись результата выполнения логической функции как состояния выхода в соответствующий разряд вектора моделирования $M(L_{is_p}) = F[M(L_{ij}) * M(L_{ir})]$.

2. ОПЕРАЦИОННОЕ УСТРОЙСТВО ДЛЯ МОДЕЛИРОВАНИЯ КОМБИНАЦИОННОЙ СТРУКТУРЫ

Процесс обработки всех примитивов схемы в данном случае является строго последовательным, что представляет собой существенное замедление процедуры формирования состояний выходных переменных. Однако уменьшение быстродействия можно считать платой за сервис встроенного и автономного восстановления работоспособности цифровой структуры, который является одним из этапов функционирования инфраструктуры обслуживания SoC, представленной на рис. 4. Комбинационная схема становится операционным устройством, где присутствуют операционный и управляющий автоматы. Заменяемыми компонентами в операционном автомате являются типы примитивов – функциональные элементы (рис. 5).

Операционное устройство реализации элементо-адресуемых комбинационных схем содержит: счетчик обработки текущего примитива $C1$; память для хранения типов примитивов, соответствующих структурным элементам P; счетчик считывания номеров входных и выходной переменных текущего примитива $C2$; дешифратор типов примитивов DC; память для хранения вектора моделирования M; матричная память для хранения номеров входов-выходов структурных примитивов L; линейка памятей, реализующих функциональные примитивы F; регистр формирования входного адресного слова для обрабатываемого примитива RG; логический элемент Or для коммутации результатов обработки функциональных примитивов.

Граф-схема алгоритма управления процессом моделирования структуры комбинационной схемы представлен на рис. 5.

1. Инициализация (формирование) всех компонентов (номера и типы элементов, линии связей для входов и выходов логических элементов) схемной структуры:

$$P = (P_1, P_2, ..., P_i, ..., P_n);\ F = (F_1, F_2, ..., F_i, ..., F_m);$$

$$L = [L_{pq}];\ p = \overline{1,n};\ q = \overline{1,s_p}.$$

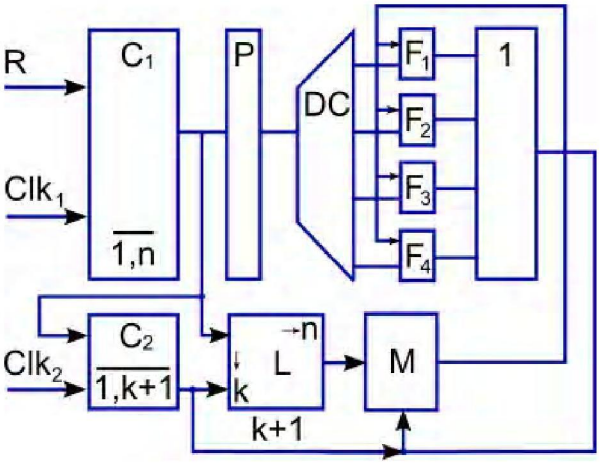

Рис. 4. Операционная структура комбинационной схемы

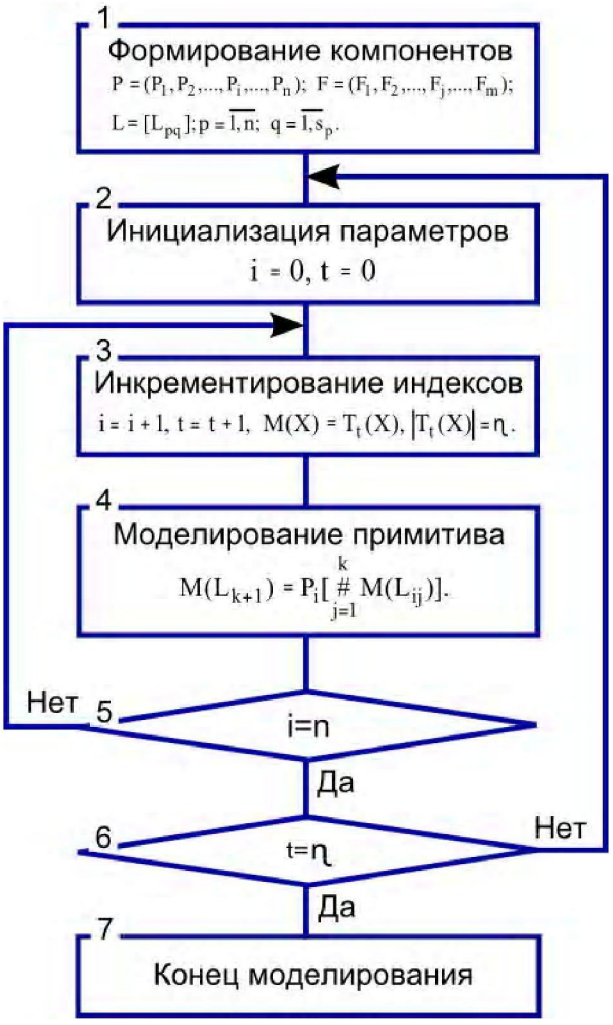

Рис. 5. Граф-схема алгоритма управления процессом моделирования

2. Инициализация параметра обрабатываемого примитива и номера входного набора $i = 0, t = 0$ для его моделирования в двоичном алфавите $M_r = \{0,1\}$.

3. Инкрементирование индекса примитива, номера теста и инициализация входного тестового (рабочего) набора: $i = i+1,\ t = t+1,\ M(X) = T_t(X),\ |T_t(X)| = \eta$.

4. Конкатенация (#) разрядов слова для формирования входного воздействия $\overset{k}{\underset{j=1}{\#}} M(L_{ij})$ логического элемента P_i и выполнение процедуры определения состояния его выхода с последующей записью в соответствующую координату вектора моделирования: $M(L_{k+1})$:

$$M(L_{k+1}) = P_i[\overset{k}{\underset{j=1}{\#}} M(L_{ij})].$$

5. Повторение пунктов 3 и 4 в целях получения состояний выходов всех логических элементов до выполнения условия: $i = n$.

6. Повторение пунктов 2–4 в целях моделирования всех входных тестовых (рабочих) наборов, до выполнения равенства: $t = \eta$, где η – длина теста.

7. Окончание процесса моделирования цифрового устройства.

ЗАКЛЮЧЕНИЕ

Научная новизна. Предложенные операционный и управляющий автоматы моделирования цифровых комбинационных схем ориентированы на решение двух практически полезных задач: 1) встроенное восстановление работоспособности компонентов комбинационных логических схем за счет увеличения времени обработки цифрового устройства и дополнительных аппаратных затрат для создания инфраструктуры моделирования адресных элементов; 2) аппаратное моделирование функциональностей цифровых проектов на основе использования PLD, что дает возможность существенно повысить быстродействие верификации программных моделей.

Практическая значимость. Положительное решение задачи встроенного ремонта логических элементов комбинационных схем дает возможность удовлетворительно решить проблему автономного восстановления работоспособности цифровых систем на кристаллах за счет временной и аппаратной избыточности проекта, что оказывает положительное влияние на надежность системы и время ее жизненного цикла.

Направления дальнейших научных исследований в данной области связаны с получением следующих результатов: 1) модели коммутирования примитивов, вышедших из строя; 2) распараллеливание вычислений по уровням элементов комбинационной схемы; 3) замена, как типов, так и примитивов структуры; 4) создание операционного устройства или инфраструктуры для моделирования последовательностных элементов и структур; 5) моделирование схем, составленных из функционально сложных примитивов; 6) разработка инфраструктуры встроенного тестирования, диагностирования и ремонта элементов комбинационных и последовательностных устройств; 7) тестирование и ремонт инфраструктуры сервисного обслуживания комбинационной схемы – решение проблемы «сторож над сторожем»; 8) Создание аналитических моделей оценки эффективности использования инфраструктуры встроенного ремонта для сервисного обслуживания комбинационных цифровых систем с различным уровнем сложности примитивов и структур.

СПИСОК ЛИТЕРАТУРЫ

1. Надежность технических систем / Под ред. И. А. Ушакова. – М. : 1985. – 512 с.
2. *Mark Gregory, Whitney.* Practical Fault Tolerance for Quantum Circuits. PhD Dissertation in Computer Science / Mark Gregory. – Berkeley : University of California, 2009. – 206 p.
3. Логический ассоциативный вычислитель / В. И. Хаханов, Е. И. Литвинова, С. В. Чумаченко [та ін.] // Электронное моделирование. – 2011. – № 1. – С. 73–90.
4. Information analysis infrastructure for diagnosis. Information. / V. Hahanov, Gharibi Wajeb, E. Litvinova, S. Chumachenko / / An international interdisciplinary journal. – Japan, 2011. – Vol. 14, No 7. – P. 2419–2433.
5. *Хаханов, В.И.* Проектирование и тестирование цифровых систем на кристаллах / В. И. Хаханов, Е. И. Литвинова, О. А. Гузь. – Харьков: ХНУРЭ, 2009.– 484 с.
6. *Hahanov, V. I.* and others. Infrastructure of intellectual property for SoC simulation and diagnosis service / Hahanov V. I. and others. – Springer : Germany, 2011. – P. 289–330.
7. *Chung, J.* Built-In Repair Analyzer With Optimal Repair Rate for Word-Oriented Memories / J. Chung, J. Park, J. A. Abraham // IEEE Transaction on Very Large Scale Integration (VLSI) Systems, 2012.– Iss. 99.– P. 1–11.
8. *Mincent, Lee.* A Memory Built-In Self-Repair Scheme Based on Configurable Spares / Lee Mincent, Denq Li-Ming, Wu Cheng-Wen // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – Vol. 30, Iss. 6. – 2011.– P. 919–929.
9. *Koal, T.* A comprehensive scheme for logic self repair / T. Koal, D. Scheit, H.T. Vierhaus // Conference Proc. on Signal Processing Algorithms, Architectures, Arrangements, and Applications.– 2009.– P. 13–18.
10. *Rab, M. T.* Improving Memory Repair by Selective Row Partitioning / M. T. Rab, A. A. Bawa, N. A. Touba // IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems.– 2009.– P. 211–219.
11. *Pekmestzi, K.* A BISR Architecture for Embedded Memories / K. Pekmestzi, N. Axelos, I. Sideris, N. Moshopoulos // 14th IEEE International On-Line Testing Symposium. – 2008. – P. 149–154.
12. *Tsu-Wei, Tseng* ReBISR: A Reconfigurable Built-In Self-Repair Scheme for Random Access Memories in SOCs / Tsu-Wei Tseng; Jin-Fu Li; Chih-Chiang Hsu // IEEE Transactions on Very Large Scale Integration (VLSI) Systems.– Vol. 18, Iss. 6.– 2010.– P. 921–932.
13. *Sharma, R. K.* Modeling and Simulation of Multi-operation Microcode-Based Built-In Self Test for Memory Fault Detection and Repair / R. K. Sharma, A. Sood // IEEE Computer Society Annual Symposium on VLSI (ISVLSI).– 2010.– P. 381–386.
14. *Oehler, P.* A Modular Memory BIST for Optimized Memory Repair / P. Oehler, A. Bosio; G. Di Natale, S. Hellebrand // 14th IEEE International On-Line Testing Symposium.– 2008.– P. 171–172.

15. *Nicolaidis, M.* Design for test and reliability in ultimate CMOS / Michael Nicolaidis, Lorena Anghel, Nacer-Eddine Zergainoh, Yervant Zorian, Tanay Karnik, Keith Bowman, James Tschanz, Shih-Lien Lu, Carlos Tokunaga, Arijit Raychowdhury, Muhammad Khellah, Jaydeep Kulkarni, Vivek De, Dimiter Avresky // Design, Automation & Test in Europe Conference & Exhibition (DATE). – 2012. – P. 677–682.
16. *Darbinyan, K.* A Robust Solution for Embedded Memory Test and Repair / K. Darbinyan, G. Harutyunyan, S.Shoukourian, V. Vardanian, Y. Zorian // 20 th Asian Test Symposium (ATS).– 2011.– P. 461–462.
17. *Grigoryan, H.* Generic BIST architecture for testing of content addressable memories / H. Grigoryan, G. Harutyunyan, S. Shoukourian, V. Vardanian, Y. Zorian // IEEE 17th International On-Line Testing Symposium (IOLTS). – 2011. – P. 86–91.
18. *Zorian, Y.* Test and reliability concerns for 3D-ICs / Y. Zorian // IEEE 16th Internationa On-Line Testing Symposium (IOLTS). – 2010. – P. 219.
19. *Koren, I.* Fault-tolerant systems. Elsevier / Koren I. and Krishna C. M. – Morgan Kaufmann Publishers, 2007. – 512 p.
20. *Koal, T.* Optimal Spare Utilization for Reliability and Mean Lifetime Improvement of Logic Built-In Self-Repair / T. Koal, H. T. Vierhaus // 14 th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). – 2011. – P. 219–224.
21. *Koal, T.* A Concept for Logic Self Repair / T. Koal, D. Scheit, H.T. Vierhaus // 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD '09. – 2009. – P. 621–624.
22. *Koal, T.* Basic Architecture for Logic Self Repair / T. Koal, H. T. Vierhaus // 14 th IEEE International On-Line Testing Symposium, IOLTS '08. – 2008. – P. 177–178.
23. *Koal, T.* A software-based self-test and hardware reconfiguration solution for VLIW processors / T. Koal, H. T. Vierhaus // IEEE Design and Diagnostics of Electronic Circuits and Systems. Los Alamitos, CA, USA. – 2010. – P. 40–43.

Стаття надійшла до редакції 19.04.2012.

Хаханов В. І., Мурад Алі Аббас, Литвинова Є. І., Хаханова І. В.

МОДЕЛІ УБУДОВАНОГО РЕМОНТУ ЛОГІЧНИХ БЛОКІВ

Запропоновано моделі комбінаційних схем, орієнтовані на розв'язання практичних задач убудованого відновлення працездатності компонентів логічних пристроїв. Логічна схема доповнюється операційним і керуючим автоматами моделювання цифрових пристроїв, що збільшує час обробки та апаратні витрати для створення оболонки елементів, що адресуються. Структури також можна використовувати для апаратного моделювання функціональностей цифрових проектів на основі використання PLD, що дає можливість суттєво підвищити швидкодію верифікації програмних моделей. Запропоноване рішення задачі вбудованого ремонту логічних елементів комбінаційних схем дає можливість комплексно вирішувати проблему автономного відновлення працездатності цифрових систем на кристалах за рахунок часової та апаратної надлишковості проекту.

Ключові слова: комбінаційна схема, відновлення працездатності, моделювання, верифікація, програмна модель.

Hahanov V. I., Murad Ali Abbas, Litvinova E. I., Hahanova I. V.

MODELS FOR EMBEDDED REPAIRING LOGIC BLOCKS

The goal of this article is to improve the quality and reliability of digital systems-on-chips by creating an infrastructure for embedded testing, diagnosis, optimization and repairing through the use of hardware redundancy and reduce the speed of functional operations. The models of combinational circuits, focused on solving real-world problems of embedded repairing components of the logic devices, are proposed. The logical scheme is improved by using operational and control automaton for modeling digital devices, and focused on solving practical problems of embedded repairing logic components by increasing the processing time and additional hardware costs to create wrapper of addressable elements. The proposed structures can also be used for hardware modeling functionalities of digital projects through the use of PLD, which allows significantly improving the performance of software model verification. The proposed solution of embedded repair of gates for combinational circuits makes it possible to comprehensively solve the problem of autonomous repair of digital system-on-chip through the use of time and hardware design redundancy.

Key words: combinational circuit, repair, modeling, verification, software model.

REFERENCES

1. Ushakov I.A. Editor Reliability of technical systems, Moscow, 1985, 512 p.
2. Mark Gregory Whitney. Practical Fault Tolerance for Quantum Circuits. PhD Dissertation in Computer Science. Berkeley, University of California, 2009, 206 p.
3. Hahanov V. I., Litvinova E. I., Chumachenko S. V., Guz O. A. Logic associative computer, *Electronic modeling*, № 1, 2011, pp. 73–90.
4. Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S. Information analysis infrastructure for diagnosis, *Information. An international interdisciplinary journal*, 2011, Japan, Vol.14, No 7, pp. 2419–2433.
5. Hahanov V. I., Litvinova E. I., Guz O. A. Digital System-on-Chip Design and Test. Kharkov, Novoye Slovo, 2009, 484 p.
6. Hahanov V. I. and others. Infrastructure of intellectual property for SoC simulation and diagnosis service. Springer, Germany, 2011, pp. 289–330.
7. Chung J., Park J., Abraham J. A. Built-In Repair Analyzer With Optimal Repair Rate for Word-Oriented Memories, *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, 2012, Issue 99, pp. 1–11.
8. Mincent Lee. A Memory Built-In Self-Repair Scheme Based on Configurable Spares / Lee Mincent, Denq Li-Ming, Wu Cheng-Wen, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 30, Issue 6, 2011, pp. 919–929.
9. Koal T., Scheit D., Vierhaus H. T. A comprehensive scheme for logic self repair, *Conference Proc. on Signal Processing Algorithms, Architectures, Arrangements, and Applications*, 2009, pp. 13–18.
10. Rab M. T., Bawa A. A., Touba N. A. Improving Memory Repair by Selective Row Partitioning, *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2009, pp. 211–219.
11. Pekmestzi K., Axelos N, Sideris I., Moshopoulos N. A BISR Architecture for Embedded Memories, *14th IEEE International On-Line Testing Symposium*, 2008, pp. 149–154.
12. Tsu-Wei Tseng, Jin-Fu Li; Chih-Chiang Hsu ReBISR: A Reconfigurable Built-In Self-Repair Scheme for Random Access Memories in SOCs, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 18, Issue 6, 2010, pp. 921–932.

13. Sharma R. K., Sood A. Modeling and Simulation of Multi-operation Microcode-Based Built-In Self Test for Memory Fault Detection and Repair, *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 381–386.
14. Oehler P., Bosio A. Di Natale G., Hellebrand S. A Modular Memory BIST for Optimized Memory Repair, *14th IEEE International On-Line Testing Symposium*, 2008, pp. 171–172.
15. Nicolaidis Michael, Anghel Lorena, Zergainoh Nacer-Eddine, Zorian Yervant, Karnik Tanay, Bowman Keith, Tschanz James, Lu Shih-Lien, Tokunaga Carlos, Raychowdhury Arijit, Khellah Muhammad, Kulkarni Jaydeep, De Vivek, Avresky Dimiter Design for test and reliability in ultimate CMOS, *Design, Automation & Test in Europe Conference & Exhibition (DATE),* 2012, pp. 677–682.
16. Darbinyan K., Harutyunyan G., Shoukourian S., Vardanian V., Zorian Y. A Robust Solution for Embedded Memory Test and Repair, *20th Asian Test Symposium (ATS),* 2011, pp. 461–462.
17. Grigoryan H., Harutyunyan G., Shoukourian S., Vardanian V., Zorian Y. Generic BIST architecture for testing of content addressable memories, *IEEE 17th International On-Line Testing Symposium (IOLTS),* 2011, pp. 86–91.
18. Zorian Y. Test and reliability concerns for 3D-ICs, *IEEE 16th Internationa On-Line Testing Symposium (IOLTS),* 2010, p. 219.
19. Koren I. and Krishna C. M. Fault-tolerant systems. Elsevier. Morgan Kaufmann Publishers, 2007, 512 p.
20. Koal T., Vierhaus H. T. Optimal Spare Utilization for Reliability and Mean Lifetime Improvement of Logic Built-In Self-Repair, *14th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2011, pp. 219–224.
21. Koal T., Scheit D., Vierhaus H. T. A Concept for Logic Self Repair, *12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD '09*, 2009, pp. 621–624.
22. Koal T., Vierhaus H.T. Basic Architecture for Logic Self Repair, *14th IEEE International On-Line Testing Symposium, IOLTS '08*, 2008, pp. 177–178.
23. Koal T., Vierhaus H. T. A software-based self-test and hardware reconfiguration solution for VLIW processors, *IEEE Design and Diagnostics of Electronic Circuits and Systems. Los Alamitos, CA, USA,* 2010, pp. 40–43.