УДК 658.512.011:681.326:519.713

Модели и методы диагностирования цифровых систем на кристаллах

В.И. Хаханов[1], Е.И. Литвинова[1], О.А. Гузь[2], С.В. Чумаченко[1]

[1]Харьковский национальный университет радиоэлектроники, hahanov@kture.kharkov.ua

[2]Донецкая академия автомобильного транспорта

***Аннотация* — Предлагается инфраструктура верификации и сервисного обслуживания цифровых систем на кристаллах на основе параллельного анализа табличных или матричных структур данных в векторном логическом пространстве при использовании мультипроцессорных архитектур. Рассматриваются модели и методы верификации, встроенного диагностирования и восстановления работоспособности компонентов цифровых систем, где качество решения оценивается неарифметической метрикой взаимодействия булевых векторов.**

***Ключевые слова* — мультипроцессор, векторно-логический анализ и критерий качества, верификация и диагностирование цифровых компонентов, процесс-модель поиска дефектов.**

I. Введение

Идея исследования – убрать из вычислителя тестирования и верификации «тяжеловесную» арифметику и направить освободившиеся ресурсы на создание инфраструктуры векторно-логических вычислений, ориентированной на эффективное тестирование и диагностирование функциональных нарушений путем анализа информационного пространства с помощью примитивных операций: and, or, not, xor. Специализация компьютерного изделия, ориентированная на использование только логических операций, дает возможность существенно (x100) повысить быстродействие решения неарифметических задач, связанных с технической диагностикой цифровых систем на кристаллах. Исключение арифметических операций, использование параллелизма алгебры векторной логики, мультипроцессорность архитектуры создают эффективную инфраструктуру, которая объединяет математическую и технологическую культуру для решения прикладных задач.

Цель исследования – существенное повышение быстродействия процедур тестирования, верификации и диагностирования путем мультипроцессорной и параллельной реализации ассоциативно-логических векторных операций для анализа графовых и табличных структур данных в векторном логическом пространстве без использования арифметических операций.

Для достижения поставленной цели необходимо разработать: 1) неарифметическую метрику оценивания векторно-логических решений в кибернетическом пространстве; 2) структуры данных и процесс-модели решения задач тестирования, верификации и диагностирования; 3) архитектуру логического ассоциативного мультипроцессора и показать пути его практического использования.

Объектом исследования является инфраструктура тестирования, верификации и диагностирования в векторно-логическом пространстве цифровых систем на кристаллах с помощью использования алгебры векторной логики, вычислительной архитектуры анализа ассоциативно-логических структур данных и неарифметического интегрального критерия качества. В процессе исследований использованы источники информации: ассоциативно-логические структуры данных для решения информационных задач [1-5]; аппаратная платформа векторно-логического анализа информации [6-9]; модели и методы тестирования, верификации и диагностирования объектов киберпространства [10-16].

II. Метрика киберпространства для оценивания решения

Дискретное векторно-логическое пространство – киберпространство – совокупность взаимодействующих по соответствующей метрике информационных процессов и явлений, описываемых векторами (кортежами) логических переменных и использующих в качестве носителя компьютерные системы и сети.

Метрика – способ измерения расстояния в пространстве между компонентами процессов или явлений, описанных векторами логических переменных. Расстояние в киберпространстве это – xor-отношение между парой многозначных (двоичных) векторов, обозначающих компоненты процесса или явления. Расстояние, производная (булева), степень изменения, различия или близости есть изоморфные понятия, связанные с определением отношения двух компонентов процесса или явления. Понятие близости (расстояния) компонентов в киберпространстве есть мера их различия. Процедуры сравнения, измерения, оценивания, распознавания, тестирования, диагностирования, идентифицирования есть способ определения xor-отношения при наличии не менее чем двух объектов. Компонент пространства представлен k-мерным (двоичным) вектором $a=(a_1,a_2,...,a_j,...,a_k)$, $a_j \in \{0,1\}$, где каждая его координата определена в двоичном алфавите: 0 – «ложь», 1 – «истина». Нуль-вектор есть k-мерный кортеж, все координаты которого равны нулю: $a_j=0,\ j=\overline{1,k}$. Метрика β кибернетического пространства определяется единственным равенством (1), которое формирует нуль-вектор для xor-суммы расстояний d_i между ненулевым и конечным числом точек (объектов), замкнутых в цикл:

$$\beta = \mathop{\oplus}\limits_{i=1}^{n} d_i = 0, \tag{1}$$

где n – количество (целое число) расстояний между компонентами (векторами) пространства, составляющими цикл $D=(d_1,d_2,...,d_i,...,d_n)$, d_i – есть вектор расстояния, соответствующий ребру цикла, соединяющему два компонента (вектора) a, b пространства, который далее обозначается без индекса как $d(a,b)$. Расстояние между двумя объектами (векторами) a и b есть производный вектор: $d(a,b)=(a_j \oplus b_j)_1^k$. Векторному значению расстояния соответствует норма – скалярное расстояние по Хэммингу между двумя векторами как число единиц вектора $d(a,b)$. Иначе: метрика β векторного логического двоичного пространства есть равная нуль-вектору xor-сумма расстояний между конечным числом точек (вершин) графа, образующих цикл. Сумма n-мерных двоичных векторов, задающих координаты точек циклической фигуры, равна нуль-вектору. Классическое задание метрики для определения взаимодействия одной, двух и трех точек в векторном логическом пространстве, является частным случаем β -метрики при $i=1,2,3$ соответственно:

$$M = \begin{cases} d_1 = 0 \leftrightarrow a = b; \\ d_1 \oplus d_2 = 0 \leftrightarrow d(a,b) = d(b,a); \\ d_1 \oplus d_2 \oplus d_3 = 0 \leftrightarrow d(a,b) \oplus d(b,c) = d(a,c). \end{cases} \tag{2}$$

Векторно-логический транзитивный треугольник (2) имеет полную аналогию численному измерению расстояния в метрическом M-пространстве, которое задается системой аксиом, определяющей взаимодействие одной, двух и трех точек в любом пространстве:

$$M = \begin{cases} d(a,b) = 0 \leftrightarrow a = b; \\ d(a,b) = d(b,a); \\ d(a,b) + d(b,c) \ge d(a,c). \end{cases} \tag{3}$$

Специфика аксиомы треугольника (3) метрического пространства заключается в численном (скалярном) сравнении расстояний трех объектов, когда интервальная неопределенность ответа – две стороны треугольника могут быть больше либо равны третьей – малопригодна для определения точной длины последней стороны. Векторно-логическое пространство устраняет данный недостаток, полностью исключает степень неопределенности в бинарном отношении детерминированных состояний процессов или явлений. В этом случае численная неопределенность третьей стороны треугольника в векторном логическом пространстве приобретает форму точного двоичного вектора, который характеризует расстояние между двумя объектами и вычисляется на основе знания расстояний двух других сторон треугольника:

$$d(a,b) \oplus d(b,c) = d(a,c) \rightarrow d(a,b) \oplus d(b,c) \oplus d(a,c) = 0.$$

Пример. Имеется пять точек в векторном пространстве: (000111, 111000, 101010, 010101, 110011). Замыкание этих точек в цикл дает следующие стороны-расстояния в пятиугольнике: (111111, 010010, 111111, 100110, 110100). Покоординатное сложение всех векторов дает результат: (000000). Практическая значимость данного факта заключается в возможности восстановления любого расстояния в замкнутом цикле, если известны (n-1) сторона фигуры. Для треугольника это означает восстановление третьей стороны по известным двум. Если же создать из треугольников замкнутое логическое пространство (рис. 1), то можно сэкономить 66% от объема данных, который формирует все расстояния в логическом пространстве.

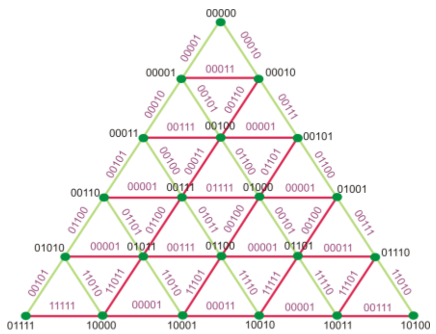


Рис. 1. Triangle Cyber Space

Метрика β кибернетического многозначного векторно-логического пространства, есть вектор, равный значению ∅ по всем координатам, полученный путем применения симметрической разности расстояний между конечным числом точек, образующих цикл:

$$\beta = \overset{n}{\underset{i=1}{\Delta}} d_i = \varnothing, \qquad (4)$$

где каждая координата вектора, соответствующего объекту, определена в алфавите, составляющем булеан на универсуме примитивов мощностью p: $a_j = \{\alpha_1, \alpha_2, ..., \alpha_r, ..., \alpha_m\}, m = 2^p$. Равенство пустому вектору симметрической разности покоординатного теоретико-множественного взаимодействия (4) подчеркивает равнозначность компонентов (расстояний), участвующих в формировании уравнения, где единственная координатная операция $d_{i,j} \Delta d_{i+1,j}$, используемая в четырехзначной модели Кантора $A = \{0, 1, x, \varnothing\}, x = \{0,1\}$, определяется как:

Δ	0	1	x	∅
0	∅	x	1	0
1	x	∅	0	1
x	1	0	∅	x
∅	0	1	x	∅

∩	0	1	x	∅
0	0	∅	0	∅
1	∅	1	1	∅
x	0	1	x	∅
∅	∅	∅	∅	∅

∪	0	1	x	∅
0	0	x	x	0
1	x	1	x	1
x	x	x	x	x
∅	0	1	x	∅

(5)

Здесь также приведены таблицы истинности для других базовых теоретико-множественных операций (пересечение, объединение, дополнение), далее используемых по тексту. Число примитивных символов, образующих замкнутый относительно теоретико-множественных координатных операций алфавит, может быть увеличено. При этом мощность алфавита (булеана) определяется выражением $m = 2^p$, где p – число примитивов. Для практического использования введенной метрики киберпространства далее предлагается доказательный переход от численной характеристики бинарного отношения объектов, объединяющей три скалярные оценки их взаимодействия к чисто векторно-логическому критерию качества отношения двух объектов. Входной вектор $m = (m_1, m_2, ..., m_j, ..., m_k),\ m_j \in \{0,1,x\}$ и объект $A = (A_1, A_2, ..., A_j, ..., A_k),\ A_j \in \{0,1,x\}$, который также представлен вектором, имеют одинаковую размерность k. Степень принадлежности m-вектора к A обозначается как $\mu(m \in A)$. Существует 5 типов координатного теоретико-множественного Δ-взаимодействия двух векторов, рис. 2: 1) $m = A$; 2) $m \subset A$; 3) $A \subset m$; 4) $m \cap A \neq \{m, A, \varnothing\}$; 5) $m \cap A = \varnothing$.

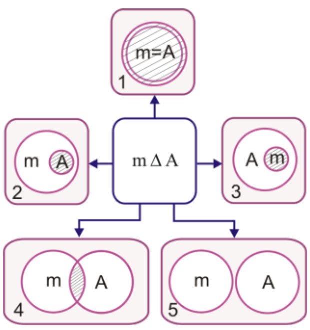


Рис. 2. Результаты взаимодействия двух векторов

Они формируют все примитивные варианты реакции системы тестирования и диагностирования на входной вектор-запрос при поиске дефектов, их распознавании, принятии решения на восстановление работоспособности. Данные стадии технологического маршрута нуждаются в метрике оценивания решений для выбора оптимального варианта. Интегральная теоретико-множественная метрика [16] для оценивания качества запроса есть функция взаимодействия многозначных по координатам векторов $m \Delta A$, которая определяется средней суммой трех параметров: кодовое расстояние $d(m, A)$, функция принадлежности $\mu(m \in A)$ и функция принадлежности $\mu(A \in m)$:

$$\begin{aligned}
&Q = (1/3)[d(m, A) + \mu(m \in A) + \mu(A \in m)],\\
&d(m, A) = (1/n)[n - \mathrm{card}\,[(i : m_i \cap A_i = \varnothing, i = \overline{1,k})]];\\
&\mu(m \in A) = 2^{c-a}; \mu(A \in m) = 2^{c-b};\\
&a = \mathrm{card}\,[(i : A_i = x, i = 1, \ldots, k)];\\
&b = \mathrm{card}\,[(i : m_i = x, i = 1, \ldots, k)];\\
&c = \mathrm{card}\,[(i : m_i \cap A_i = x, i = 1, \ldots, k)];
\end{aligned} \qquad (6)$$

Пересечение (объединение) векторов – есть векторная операция, основанная на соответствующих координатных теоретико-множественных операциях.

Операции координатного пересечения и объединения (6) определены в алфавите Кантора $A = \{0, 1, x = \{0,1\}, \varnothing\}$. Нормирование параметров позволяет оценить уровень взаимодействия векторов в численном интервале [0,1]. Если зафиксировано предельное максимальное значение каждого параметра, равное 1, то векторы равны между собой. Минимальная оценка $Q = 0$ фиксируется в случае полного несовпадения векторов по всем n координатам. Если $m \cap A = m$ и мощность покоординатного пересечения равна половине мощности пространства вектора A, то функции принадлежности и качества равны:

$$\mu(m \in A) = \frac{1}{2};\ \mu(A \in m) = 1;\ d(m, A) = 1;\ \ Q(m, A) = \frac{5}{6}.$$

Аналогичное значение будет иметь параметр Q, если $m \cap A = A$ и мощность покоординатного пересечения равна половине мощности пространства вектора m. Здесь пространство вектора есть функция от числа координат ω, равных x: $q = 2^{\omega}$. Например, даны два вектора: A = (XXX10) и m = (XX0X0). Их пересечение равно (XX010) = {00010, 01010, 10010, 11010}. Иначе, мощность результирующего пространства равна четырем двоичным векторам или половине мощностей исходных двоичных векторов. Следует заметить, если пересечение двух векторов равно пустому множеству $\exists i(m_i \cap A_i) = \varnothing$, то количество общих точек (двоичных векторов) при пересечении двух пространств, формируемых двумя векторами, равно нулю. С учетом изоморфизма теоретико-множественных и логических операций арифметический критерий (6) без усреднения функций принадлежности и кодового расстояния можно трансформировать к виду:

$$\begin{aligned}
&Q = d(m, A) + \mu(m \in A) + \mu(A \in m),\\
&d(m, A) = \mathrm{card}\left(\{i : m_i \oplus A_i = U, i = 1,\ldots,k\}\right);\\
&\mu(m \in A) = \mathrm{card}\ (\{i : A_i = U, i = 1,\ldots,k\}) -\\
&- \mathrm{card}\left(\{i : m_i \oplus A_i = U, i = 1,\ldots,k\}\right);\\
&\mu(A \in m) = \mathrm{card}\,(\{i : m_i = U, i = 1,\ldots,k\}) -\\
&- \mathrm{card}\left(\{i : m_i \oplus A_i = U, i = 1,\ldots,k\}\right);
\end{aligned} \qquad (7)$$

$$U = \begin{cases} 1 \leftarrow \{m_i, A_i\} \in \{0,1\}; \\ x \leftarrow \{m_i, A_i\} \in \{0,1,x\}. \end{cases}$$

Если векторы m и A – двоичные по всем координатам, то переменная U=1 и вычисления проводятся по правилам двоичной $\oplus$ –операции. Если векторы m и A определены в троичном алфавите, то переменная U=x инициирует вычисления на основе использования теоретико-множественной операции симметрической разности Δ. Первый компонент (7) формирует степень несовпадения k-мерных векторов – кодовое расстояние, путем выполнения операции xor, второй и третий определяют степени непринадлежности результата конъюнкции к числу единиц каждого из двух взаимодействующих векторов. Понятия принадлежности и непринадлежности являются взаимодополняющими, но в данном случае технологичнее вычислять непринадлежность. Для того, чтобы окончательно исключить арифметические операции при подсчете векторно-логического критерия качества, необходимо логически объединить три оценки (7) в одну [16]:

$$Q = d(m, A) \vee \mu(m \in A) \vee \mu(A \in m) = m \oplus A.$$

Процедура вычисления векторного критерия качества зависит от значности алфавита:

$$Q' = \begin{cases} m \oplus A \leftarrow \{m_i, A_i\} \in \{0,1\}; \\ m\Delta A \leftarrow \{m_i, A_i\} \in \{0,1,x\}. \end{cases} \qquad (8)$$

Критерий качества однозначно определяет три формы взаимодействия двух любых объектов в n-мерном векторном логическом пространстве: расстояние и две функции принадлежности.

Для сравнения критериев качества необходимо определять число единиц в каждом векторе без выполнения операций суммирования. Это можно сделать с помощью регистра сдвига [6], который позволяет за один такт выполнить процедуру slc (shift left bit crowding) – сдвиг влево с одновременным уплотнением всех единиц n-разрядного двоичного вектора (рис. 3).

Процесс-модель поиска оценки лучшего решения с минимальным числом единичных координат из более чем двух альтернатив представлена на рис. 4. Она включает следующие операции: 1) первоначально в вектор-результат Q, в котором будет сохранено лучшее решение, заносятся единичные значения во все координаты (худшее решение) и одновременно осуществляется операция slc сдвига влево с уплотнением единиц текущего вектора Q_i.

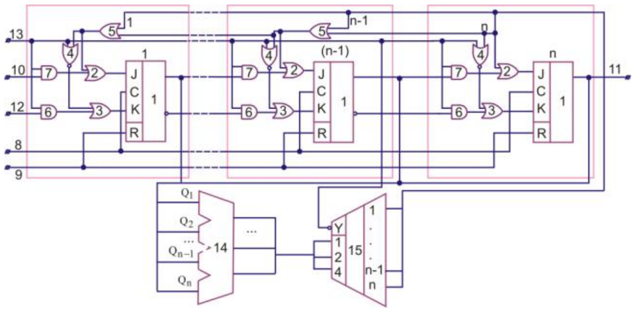


Рис. 3. Регистр сдвига и уплотнения единиц

2) Выполняется сравнение двух векторов: Q и очередной оценки Q_i из списка решений. 3) Реализуется векторная операция and $(Q \wedge Q_i)$, а результат сравнивается с вектором Q, что дает возможность изменить его, если вектор Q_i имеет меньшее число единичных значений. 4) Процедура поиска оценки лучшего решения повторяется n раз.

$$\begin{aligned}
&Q = Q\overline{(\vee((Q \wedge Q_i) \oplus Q))} \vee Q_i(\vee((Q \wedge Q_i) \oplus Q));\\
&Y = \vee((Q \wedge Q_i) \oplus Q);\\
&Q = Q\overline{Y} \vee Q_i Y.
\end{aligned}$$

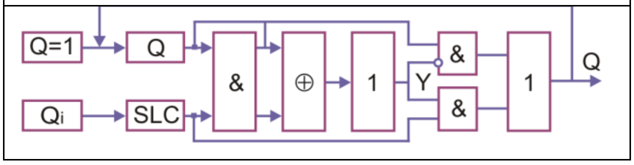


Рис. 4. Процесс-модель выбора решения

Здесь or-оператор редукции (после xor элемента) формирует двоичное однобитовое решение на основе логической операции or над n разрядами критерия качества.

III. Архитектура логического ассоциативного мультипроцессора

Для анализа больших информационных объемов логических данных предлагается использовать логический ассоциативный мультипроцессор (ЛАМП) – это эффективная сеть процессоров, которая обрабатывает данные и обеспечивает обмен информацией между компонентами сети в процессе их решения. Простая схемотехника каждого процессора позволяет эффективно обрабатывать сверхбольшие массивы, насчитывающие миллионы бит информации, затрачивая на это существенно (x100) меньше времени по сравнению с универсальным процессором.

Базовая ячейка – векторный процессор для вычислителя может быть синтезирован на 200-х вентилях, что дает возможность сеть, содержащую 4096 вычислителей, легко реализовать в кристалле заказной СБИС, используя современную кремниевую технологию. Поскольку затраты памяти для хранения данных весьма незначительны, вычислитель может быть использован при проектировании систем управления в таких областях человеческой деятельности, как промышленное производство, защита информации, медицина, искусственный интеллект, космонавтика, геология, метеорология. Основное назначение ЛАМП – получение квази-оптимального решения при верификации и диагностировании цифровых систем на основе выполнения векторных логических операций:

$$P(m, A) = \min Q_i (m \mathop{\Delta}_{i=1}^{n} A_i),\ m = \{m_a, m_b, m_c, m_d\}. \quad (9)$$

Один из возможных вариантов архитектуры ЛАМП представлен на рис. 5. Основным компонентом является матрица $P = [P_{ij}], (i.j = \overline{1,4})$, содержащая 16 вектор-процессоров, каждый из которых предназначен для выполнения пяти логических векторных операций над памятью данных, представленной в виде таблицы (матрицы) A, размерностью $(m \times n)$. Компоненты или регистры $m = (m_a, m_b, m_c, m_d)$ используются для получения решения в виде буферных, входных и выходных векторов, а также для идентификации оценки качества удовлетворения входного запроса. Блок управления инициирует выполнение команд логической обработки данных и синхронизирует функционирование всех компонентов мультипроцессора. Блок IP [8] предназначен для сервисного обслуживания всех модулей, диагностирования дефектов и восстановления работоспособности компонентов и устройства в целом. Логический ассоциативный процессор (ЛАП) (см. рис. 5), входящий в состав вычислителя, содержит логический процессор LP; ассоциативную (память) A для параллельного выполнения базовых операций; блок векторов m, предназначенный для параллельного обслуживания строк и столбцов матрицы A, а также обмена данными в процессе вычислений; память прямого доступа CM, сохраняющую команды программы обработки информации; CU – автомат управления выполнением логических операций; интерфейс I связи ЛАП с другими элементами и устройствами ЛАМП.

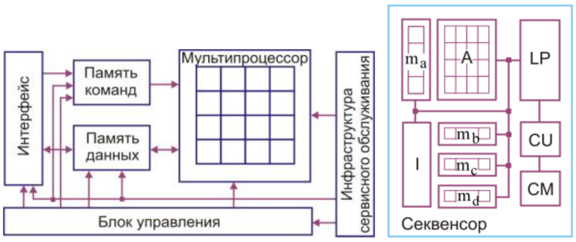


Рис. 5. Архитектура ЛАМП и структура ЛАП

Логический процессор (LP) (рис. 6) выполняет пять операций (and, or not, xor, s1c), являющихся базовыми для создания алгоритмов и процедур диагностирования. Модуль LP имеет мультиплексор, коммутирующий один из пяти операндов с выбранным логическим векторным оператором. Сформированный результат через мультиплексор (элемент or) заносится в один из четырех операндов, выбираемый соответствующим адресом. Особенности реализации логического процессора заключаются в наличии трех бинарных (and, or, xor) и двух унарных (not, slc) операций. Все операции в LP — регистровые или регистрово-матричные. Последние предназначены для анализа вектор-строк таблицы при использовании входного m-вектора как запроса для точного поиска информации. Реализация всех векторных операций блока логических вычислений, выполняемых с тактовой частотой 100 МГц, для одного ЛАП в среде Verilog с последующей послесинтезной реализацией в кристалле программируемой логики Virtex 4, фирмы Xilinx содержит 2400 эквивалентных вентилей.

IV. Инфраструктура векторно-логического анализа

Инфраструктура – совокупность моделей, методов и средств описания, анализа и синтеза структур данных для сервисного обслуживания цифровых систем на кристаллах. Для детализации структуры векторного процессора и устройства последовательного управления (УПУ) далее рассмотрены аналитические и структурные процесс-модели, выполняющие анализ A-матрицы по столбцам или строкам.

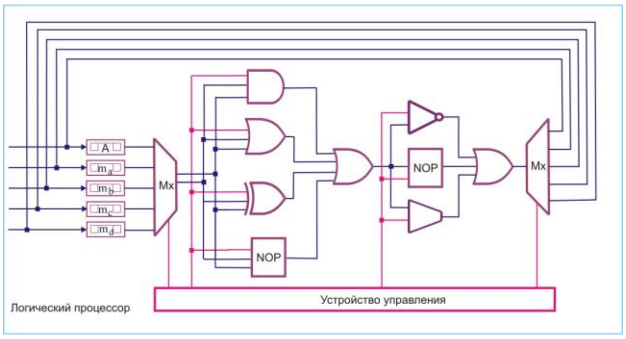


Рис. 6. Структура блока логических вычислений

Первая из них представлена на рис. 7 и предназначена для определения множества допустимых решений относительно входного запроса m_b, вторая (рис. 8) осуществляет поиск оптимального решения на множестве строк, найденных с помощью первой модели в результате их анализа.

$$m^m_{ai} = \overline{\vee} [(m_b \mathop{\wedge}_{i=1}^{n} A_i) \oplus m_b];$$

$$A_i = (m_b \mathop{\wedge}_{i=1}^{n} A_i).$$

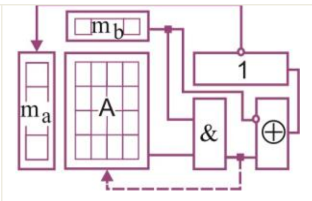


Рис. 7. Поиск всех допустимых решений

Возможно и самостоятельное применение второй модели, ориентированное на определение однозначного и многозначного решения при поиске дефектов в цифровой системе.

$$m_b^s = (\bigwedge_{\forall m_{ai}=1} A_i) \wedge (\overline{\bigvee_{\forall m_{ai}=0} A_i})$$

$$m_b^m = (\bigvee_{\forall m_{ai}=1} A_i) \wedge (\overline{\bigvee_{\forall m_{ai}=0} A_i})$$

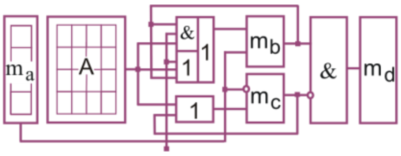


Рис. 8. Выбор оптимального решения

Все операции, выполняемые двумя процесс-моделями – векторные. Модель анализа строк (см. рис. 8) формирует вектор m_a идентификации допустимых $m_{ai} = 1$ или противоречивых $m_{ai} = 0$ решений относительно входного условия m_b за n тактов обработки всех m-разрядных векторов таблицы $A = card(m \times n)$. Качество (допустимость) решения определяется для каждого взаимодействия входного вектора m_b и строки $A_i \in A$ на блоке (редукции) дизъюнкции. Матрица A может быть модифицирована ее пересечением с входным вектором на основе использования операции $A_i = (m_b \bigwedge_{i=1}^{n} A_i)$, если необходимо исключить из A-таблицы все незначимые для решения координаты и векторы, отмеченные единичными значениями вектора m_a.

Решение задач диагностирования посредством анализа строк таблицы (см. рис. 8) осуществляется так. После выполнения диагностического эксперимента формируется двоичный вектор экспериментальной проверки m_a, маскирующий A-таблицу неисправностей для поиска одиночных или кратных дефектов. Векторы m_b и m_c используются для накопления результатов выполнения операций конъюнкции и дизъюнкции. Затем выполняется логическое вычитание (xor-операция) из первого регистра m_b содержимого второго вектора m_c с последующей записью результата в регистр m_d. Для реализации второго уравнения, которое формирует множественное решение, элемент and заменяется функцией or. В схеме используется также переменная выбора режима поиска решения: single или multiple. В качестве входного условия в модели использован вектор m_a, управляющий выбором векторной операции and или or для обработки единичных $A_i(m_{ai} = 1) \in A$ или нулевых $A_i(m_{ai} = 0) \in A$ строк A-таблицы. В результате выполнения n тактов осуществляется накопление единичных и нулевых относительно значений координат вектора m_a решений в регистрах A_1, A_0. Априори в указанные регистры заносится вектор единиц и нулей: $A_1 = 1, A_0 = 0$. После обработки всех n строк A-таблицы за n тактов выполняется векторная конъюнкция содержимого регистра A_1 с инверсией регистра A_0, которая формирует результат в виде вектора m_b, где единичные значения координат определяют решение. В таблице неисправностей цифрового изделия единичным координатам вектора m_b соответствуют столбцы, отождествляемые с номерами дефектов или неисправных блоков, подлежащих восстановлению или ремонту.

При сервисном обслуживании функциональных модулей можно на универсальной структуре системы векторного логического анализа решить оптимизационную задачу восстановления работоспособности. С помощью минимального числа ремонтных запасных строк и (или) столбцов, например, памяти, необходимо обеспечить квазиоптимальное покрытие всех обнаруженных в ячейках неисправностей. Технологическая и математическая составляющие векторной логики в данном случае обусловливают простое схемотехническое решение для получения квазиоптимального покрытия (рис. 9), преимущества которого заключается в следующем: 1. Вычислительная сложность процедуры: число векторных операций, равное числу строк таблицы, $Z = n$. 2. Минимум аппаратных затрат: таблица и два вектора (m_b, m_a) для хранения промежуточных покрытий и накопления результата в виде единичных координат, соответствующих строкам таблицы, которые составляют квазиоптимальное покрытие. 3. Отсутствие классического деления задачи покрытия на поиск ядра покрытия и дополнения. 4. Отсутствие сложных процедур манипулирования ячейками строк и столбцов. Получение не всегда оптимального покрытия — недостаток, который компенсируется технологичностью векторной процедуры, представленной на рис. 9.

$$\begin{cases} m_b = (m_b \vee A_i); \\ m_{ai} = \bigvee_{i=1}^{n} [(m_b \vee A_i) \wedge \overline{m}_b]. \end{cases}$$

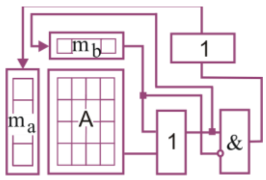


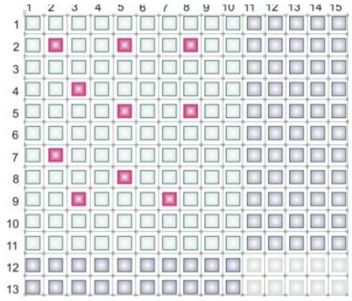
Рис. 9. Процесс-модель поиска квазиоптимального покрытия

Операция редукции на последнем этапе превращает векторный результат в бит m_{ai} вектора m_a по функции or $m_{ai} = \vee[(m_b \vee A_i) \wedge \overline{m}_b]$. В общем случае операция редукции в алгебре векторных операций записывается в виде <бинарная операция><вектор>: $\vee A_i, \wedge m, \overline{\wedge}(m \vee A_i)$. Обратная процедура – векторизация есть конкатенация булевых переменных: $m_a(a,b,c,d,e,f,g,h)$. В процедуре поиска покрытия априори векторы $m_b = 0$, $m_a = 0$ становятся равными. Квазиоптимальное покрытие накапливается за n тактов в векторе m_a последовательным сдвигом. Биты, заносимые в регистр m_a, формируются схемой or, которая выполняет редукцию после анализа полученного результата $[(m_b \vee A_i) \wedge \overline{m}_b]$ на наличие единиц.

Представляет интерес функциональная законченность цикла диагностирования, когда после получения квазиоптимального покрытия данная информация используется для восстановления работоспособности дефектных ячеек памяти [8]. Размерность модуля памяти (13x15 ячеек) не влияет на вычислительную сложность получения покрытия десяти дефектных ячеек с помощью резервных строк и столбцов (рис. 10).

Для решения оптимизационной задачи выполняется построение таблицы покрытия (см. рис. 10) неисправных ячеек, в которой строки – резервные ресурсы для полного покрытия дефектов $(C_2, C_3, C_5, C_7, C_8, C_2, R_2, R_4, R_5, R_7, R_8, R_9)$, а столбцы – соответствующие дефекты ячеек $(F_{2,2}, F_{2,5}, F_{2,8}, F_{4,3}, F_{5,5}, F_{5,8}, F_{7,2}, F_{8,5}, F_{9,3}, F_{9,7})$, подлежащие ремонту. При этом столбцы соответствуют координатам дефектных ячеек, а строки идентифицируют резервные компоненты (строки и столбцы), которые могут восстановить работоспособность неисправных координат. Модель вычислительного процесса, представленная на рис. 11, дает возможность получить оптимальное решение в виде $m_a = \boxed{1\,1\,1\,1\,1\,0\,0\,0\,0\,0}$, которому соответствует покрытие $R = \{C_2, C_3, C_5, C_7, C_8\}$ как одно из трех возможных минимальных решений $R = C_2, C_3, C_5, C_7, C_8 \vee C_2, C_3, C_5, C_8, R_9 \vee C_2, C_5, C_8, R_4, R_9$ для таблицы неисправностей. Технологическая модель встроенного диагностирования и ремонта памяти (рис. 11) имеет четыре компонента: 1. Тестирование модуля (Unit Under Test (UUT)) с использованием эталонной модели (Model Under Test (MUT)) для формирования вектора экспериментальной проверки m_a, размерность которого соответствует числу тестовых наборов. 2. Поиск дефектов на основе анализа таблицы неисправностей A. 3. Оптимизация покрытия дефектных ячеек ремонтными строками и столбцами на основе анализа таблицы A. 4. Восстановление работоспособности памяти посредством замены адресов (Address Decoder (AD)) неисправных строк и столбцов, представленных вектором m_a, на адреса компонентов из запаса (Spare Memory (SM)) [8].

Процесс-модель встроенного сервисного обслуживания работает в реальном масштабе времени и позволяет поддерживать в работоспособном состоянии, без вмешательства человека, цифровую систему на кристалле, что является целесообразным решением в случае применения технологий, связанных с дистанционной эксплуатацией изделия.



1	.	.	.	.	.	1	.	.	.
.	.	.	1	.	.	.	.	1	.
.	1	.	.	1	.	.	1	.	.
.	.	.	.	.	.	.	.	.	1
.	.	1	.	.	1	.	.	.	.
1	1	1	.	.	.	.	.	.	.
.	.	.	1	.	.	.	.	.	.
.	.	.	.	1	1	.	.	.	.
.	.	.	.	.	.	1	.	.	.
.	.	.	.	.	.	.	1	.	.
.	.	.	.	.	.	.	.	1	1

Рис. 10. Модуль памяти с резервом и таблица покрытия

Предложенные процесс-модели анализа ассоциативных таблиц, а также введенные критерии качества логических решений позволяют решать задачи квазиоптимального покрытия, диагностирования дефектов программных и (или) аппаратных блоков. Модель векторных вычислений стала основой для разработки специализированной мультипроцессорной архитектуры, ориентированной на поиск, распознавание и принятие решений об использовании структур ассоциативных таблиц.

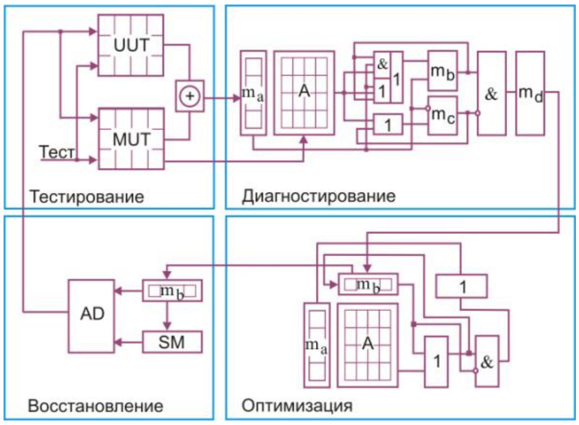


Рис. 11. Модель встроенного тестирования и восстановления памяти

Аналитическая оценка эффективности проектного решения, направленного на выполнение условий специализации Sp и стандартизации St (рис. 12) определяется минимумом среднего значения следующих трех взаимно противоречивых относительных и безразмерных параметров: уровень ошибок проекта L, время верификации и (или) тестирования T, программно-аппаратная избыточность, определяемая механизмами ассерций и (или) граничного сканирования H.

$$E = F(L, T, H) = \min\left[\frac{1}{3}(L + T + H)\right],$$
$$Y = (1 - P)^{n};$$
$$L = 1 - Y^{(1-k)} = 1 - (1 - P)^{n(1-k)};$$
$$T = \frac{(1 - k) \times H^{s}}{H^{s} + H^{a}};\ H = \frac{H^{a}}{H^{s} + H^{a}}.$$

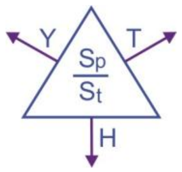


Рис. 12. Оценка эффективности процесс-модели

Параметр L, как дополнение к Y, характеризующему выход годной продукции, зависит от тестопригодности проекта k, вероятности P существования неисправных компонентов и числа необнаруженных ошибок n. Время верификации определяется тестопригодностью проекта k, умноженной на структурную сложность аппаратно-программной функциональности, отнесенной к общей сложности проекта в строках кода или эквивалентных вентилях. Программно-аппаратная избыточность находится в функциональной зависимости от сложности ассерционного кода или механизма граничного сканирования, отнесенной к общей сложности проекта. При этом ассерционная, или сканирующая, избыточность должна обеспечивать заданную глубину диагностирования ошибок функциональности за время выхода изделия на рынок, определенное заказчиком.

V. Выводы

Существующие аналоги практически не предлагают ассоциативно-логических технологий тестирования, верификации и диагностирования в дискретном пространстве функционирования цифровых систем [4, 5, 15]. Практически все они используют универсальную систему команд современного дорогостоящего процессора с математическим сопроцессором. В то же время, аппаратные специализированные средства логического анализа, являющиеся их прототипами [1, 6, 7], как правило, ориентированы на побитовую или невекторную обработку информации. Предложенный новый подход векторно-логической обработки ассоциативных данных с полным исключением арифметических операций, влияющих на быстродействие и аппаратную сложность, может быть эффективно реализован на основе использования современной микроэлектронной аппаратуры в виде мультипроцессорной цифровой системы на кристалле. Реализация подхода основана на предложении следующих моделей и методов, использующих общую идею вычисления взаимодействия объектов киберпространства с помощью векторно-логической метрики: 1. Процесс-модели анализа ассоциативных таблиц, ориентированные на достижение высокого быстродействия при подсчете критериев качества взаимодействия объектов на основе векторных логических операций для тестирования верификации и диагностирования цифровых систем. 2. Метод параллельного решения ассоциативно-логических задач с минимальным числом векторных логических операций и полным исключением арифметических команд, что обеспечивает, минимальную стоимость, время и незначительное энергопотребление вычислителя, реализованного на кристалле программируемой логики. 3. Новые векторно-логические процесс-модели встроенного диагностирования и верификации цифровых систем на кристаллах, использующие средства логического ассоциативного мультипроцессора, параллельные операции вычислительных процессов и векторно-логический критерий качества. Практическая значимость полученных результатов подтверждена созданием встроенного компонента для диагностирования и восстановления работоспособности памяти в цифровой системе на кристалле.

Литература

[1] Бондаренко М.Ф. О мозгоподобных ЭВМ / М.Ф. Бондаренко, З.В. Дударь, И.А. Ефимова, В.А. Лещинский, С.Ю. Шабанов–Кушнаренко // Радиоэлектроника и информатика. Харьков: ХНУРЭ. 2004, № 2. С. 89–105.

[2] Cohen A.A. Addressing architecture for Brain-like Massively Parallel Computers // Euromicro Symposium on Digital System Design (DSD'04). 2004. P. 594-597.

[3] Кузнецов О.П. Быстрые процессы мозга и обработка образов // Новости искусственного интеллекта. 1998. №2.

[4] Васильев С.Н., Жерлов А.К., Федосов Е.А., Федунов Б.Е. Интеллектуальное управление динамическими системами. М.: Физико-математическая литература. 2000. 352 с.

[5] Липаев В.В. Программная инженерия. Методологические основы. Учебник. М.: Теис, 2006. 608 с.

[6] АС СССР 1439682. Регистр сдвига / Какурин Н.Я., Хаханов В.И., Лобода В.Г., Какурина А.Н. ; заявл. 07.04.1987 ; опубл. 23.11.1988. 4 с.

[7] Гайдук С.М., Хаханов В.И., Обризан В.И., Каменюка Е.А. Сферический мультипроцессор PRUS для решения булевых уравнений // Радиоэлектроника и информатика. Харьков, 2004. № 4(29). С.107-116.

[8] Хаханов В.И. Проектирование и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. – Харьков: ХНУРЭ, 2009.– 484с.

[9] Хаханов В.И. Проектирование и верификация цифровых систем на кристаллах / В.И. Хаханов, И.В. Хаханова, Е.И. Литвинова, О.А. Гузь. Харьков: Новое слово. 2010. 528 с.

[10] Акритас А. Основы компьютерной алгебры с приложениями / А. Акритас. М.: Мир. 1994. 544 с.

[11] Аттетков А.В. Методы оптимизации / А.В. Аттетков, С.В. Галкин, В.С. Зарубин. М.: Издательство МГТУ им. Н.Э. Баумана. 2003. 440 с.

[12] Abramovici M. Digital System Testing and Testable Design / M. Abramovici, M.A. Breuer and A.D. Friedman. Comp. Sc. Press. 1998. 652 p.

[13] Densmore D.A Platform–Based taxonomy for ESL Design / Douglas Densmore, Roberto Passerone, Alberto Sangiovanni–Vincentelli // Design & Test of computers. 2006. P. 359-373.