УДК 658.512.011:681.326:519.713

ЛОГИЧЕСКИЙ АССОЦИАТИВНЫЙ МУЛЬТИПРОЦЕССОР ДЛЯ АНАЛИЗА ИНФОРМАЦИИ

М.Ф. Бондаренко[1], В.И. Хаханов[2]

[1]ХНУРЭ, г. Харьков, Украина,
[2]ХНУРЭ, г. Харьков, Украина, hahanov@kture.kharkov.ua

Предлагается архитектура быстродействующего мультипроцессора параллельного анализа информации, представленной в виде аналитических, графовых и табличных структур ассоциативных отношений, для поиска, распознавания и принятия решений в n-мерном векторном дискретном пространстве. Рассматриваются векторно-логические процесс-модели актуальных прикладных задач, качество решения которых оценивается введенной интегральной неарифметической метрикой взаимодействия булевых векторов.

МУЛЬТИПРОЦЕССОР, АНАЛИЗ ИНФОРМАЦИИ, ГРАФ, ТАБЛИЦА, АССОЦИАТИВНОЕ ОТНОШЕНИЕ, ПРОЦЕСС-МОДЕЛЬ.

Введение

Мозг и компьютер. 1) Сходство и различие. Технологическая основа примитивных функций мозга и компьютера – логические операции: and, or, not, xor. Различие тоже понятно – кремниевая (цифровая) и биологическая (аналоговая) природа реализации элементарных операций. 2) Функциональность. Как в мозге, так и в компьютере, «выращиваются» более сложные функциональные пространственно-временные логические преобразователи, использующие упомянутые выше примитивные операции. Логические операции лежат в основе алгоритмов компьютерного решения любой задачи: арифметической, логической или комбинированной. Однако утяжеление универсального компьютера путем возведения на базисе логических функций (and, or, not) приводит не только к расширению его функциональностей, но и к удорожанию и снижению быстродействия по отношению к конкретной задаче. Специализация компьютерного изделия, ориентированная на использование логических операций, дает возможность приблизиться к ассоциативно-логическому мышлению человека и вместе с тем существенно (x100) повысить быстродействие решения специальных задач. Исключение арифметических операций в специализированном процессоре, использование векторной логики, мультипроцессорность архитектуры для параллельного анализа информации есть старый и забытый маршрут повышения быстродействия решения прикладных задач на современной технологической основе. 3) Структурная организация. Мозгоподобность компьютера здесь рассматривается не как перенос нейроструктуры в архитектуру компьютера, а как имплементация в него ассоциативно-логической функциональности мозга. Создание компьютера с мозгоподобной структурой, выполняющей аналоговые вычисления, пока бесперспективно для эффективной реализации цифрового вычислительного устройства. Тем не менее, соединение биотехнологий и наноэлектроники уже через 5 лет может дать практически ориентированный результат в виде сети или матрицы элементарных логических процессоров (здесь структурное подобие мозгу), параллельно решающих ассоциативно-логические задачи (функциональное подобие мозгу) на информационных массивах большой размерности. Не следует также переносить идеально выверенную тысячелетней эволюцией биологическую модель мозга на несовершенную и незрелую структуру кремниевого кристалла. Такого рода мозгоподобность для решения практических задач в недалеком прошлом уже терпела неудачи. Другое дело, когда функциональности, присущие мозгу, реализуются в кристалле кремния путем соединения выверенных временем аналитических и синтетических свойств мозга с преимуществом быстродействующих цифровых платформ для решения ассоциативно-логических задач.

Определение. Мозгоподобность мультипроцессорной цифровой системы на кристалле есть концепция создания архитектуры и моделей вычислительных процессов, ориентированных на эффективную и быстродействующую реализацию функциональностей, свойственных мозгу, на основе использования векторных логических операций для решения задач поиска, распознавания и принятия решений.

1. Класс задач для мозгоподобного мультипроцессора

Существует большой класс логических задач, которые в настоящее время решаются не эффективно на универсальных компьютерах путем использования арифметических операций. Система логических команд является универсальным и полным базисом, а согласно теореме Поста способна описать и решить любую задачу. Однако практические разработчики программно-аппаратных систем уже

забыли об изначальной логической сущности компьютера. Они привыкли неэффективно (на 5-10%) использовать мощности системы команд, компиляторов, операционных систем, отдавая компаниям-производителям порядка 90% денежных средств безвозмездно при покупке компьютера. Решение логических задач с помощью арифметического процессора не есть правильный выбор с позиции технологической и математической культуры. В настоящее время существующая платформа цифровых систем на кристаллах предоставляет практически неограниченные возможности для переориентации инфраструктуры моделей и методов на создание специализированных компьютеров с минимальной системой логических векторных операций или команд. Мотивация – создание мультипроцессора, как специализированной системы на кристалле для анализа и синтеза логических ассоциативных отношений, свойственных мозгу. Подтверждением актуализации таких изделий является появление на рынке iPad планшета (толщиной 12 мм и весом 0,68 кг), процессор которого выполнен в виде системы на кристалле Apple A4 на базе многоядерного ARM-процессора Cortex-A9 MPCore с использованием контроллеров памяти и графики. Наличие аппаратного, быстродействующего и дешевого специализированного логического вычислителя позволяет эффективно решать интересные для рынка информационных технологий задачи: 1. Анализ и синтез синтаксических и семантических языковых конструкций (реферирование, исправление ошибок, оценивание качества текстов). 2. Распознавание видео- и аудио-образов путем их представления вектором существенных параметров в дискретном пространстве. 3. Сервисное обслуживание сложных технических изделий и восстановление работоспособности в процессе их функционирования. 4. Тестирование знаний и экспертное обслуживание объектов или субъектов для определения их валидности. 5. Идентификация объекта или процесса для принятия решения в условиях неопределенности. 6. Точный поиск заданной вектором параметров информации в Internet, где по запросу пользователя очень часто выдаются два сообщения: отсутствуют данные или слишком много информации, слабо ассоциируемой с входным запросом. Здесь нужна правильная метрика оценивания и валидный запрос. 7. Коррекция текста в процессе его набора, когда автоматически исправляются только тривиальные ошибки, такие как повторение буквы в слове. Можно также корректировать более сложные семантические ошибки, связанные с неверным окончанием, и предлагать более приемлемые варианты порядка слов в предложении или в его части. Данная задача актуальна для 100% пользователей компьютеров. 8. Более серьезная проблема выбора принадлежит критическим технологиям: целеуказание в истребителе или в автоматической системе посадки лайнера, работающих в реальном масштабе времени в микросекундном диапазоне измерения. 9. Обратной задачей выбора цели в критических технологиях является разведение объектов во времени и в пространстве, например, в диспетчерской службе аэропорта или оптимизация инфраструктуры городского транспорта для исключения коллизий. Практически все упомянутые задачи решаются в реальном масштабе времени, являются сходными по логической структуре процесс-моделей на основе использования ассоциативных таблиц. Для их решения необходима быстродействующая и специализированная аппаратная платформа в виде логического ассоциативного мультипроцессора (LAMP – Logical Associative MultiProcessor), ориентированного на параллельное выполнение процедур поиска, распознавания и принятия решений, оцениваемых путем использования интегрального критерия качества.

Цель – существенное повышение быстродействия процедур поиска, распознавания и принятия решений путем мультипроцессорной реализации параллельных средств обработки аналитических, графовых и табличных форм задания информации для определения детерминированного многозначного решения в n-мерном дискретном булевом пространстве.

Задачи: 1) Актуальность создания мозгоподобных вычислителей. 2) Метрика оценивания векторно-логических решений. 3) Архитектуры структур данных и ассоциативных отношений: таблицы, графы, уравнения. 4) Оптимизация логических структур данных. 5) Архитектура логического ассоциативного мультипроцессора. 6) Процесс-модели решения практически интересных задач на основе архитектуры LAMP.

Сущность – аппаратное обеспечение экспертного обслуживания запросов в реальном масштабе времени в виде мультипроцессорной системы на кристалле, ориентированной на анализ логических ассоциативных структур данных для получения точного детерминированного и многозначного решения, валидность (состоятельность) которого оценивается интегральным критерием качества взаимодействия запроса с векторами n-мерного ассоциативного пространства.

Объект исследования – аппаратная инфраструктура экспертного обслуживания задач поиска, распознавания и принятия решений в дискретном булевом пространстве на основе использования интегрального критерия качества и иерархических структур данных.

Предмет исследования – аппаратная платформа и технологии для реализации мультипроцессорной системы на кристалле, ориентированной на обслуживание задач поиска, распознавания и выбора решения в ассоциативных информацион-

ных структурах путем использования интегрального критерия качества в дискретном булевом пространстве.

Источники. 1. Аппаратные вычислительные изделия, ориентированные на решение логических задач ассоциативного поиска [1-4]. 2. Ассоциативные и логические структуры данных для решения информационных задач [5-8]. 3. Модели и методы дискретного анализа информации [9-12]. 4. Мультипроцессорные модели и средства для решения информационно-логических задач [13-19].

2. Интегральная метрика оценивания решения

При создании аппаратной платформы для информационно-логических задач акцент делается на следующие характеристики: 1) Высокое быстродействие параллельного выполнения минимального множества логических команд. 2) Исключение из процессора мощной системы арифметических вычислений, как функциональностей, несвойственных человеку. 3) Логический секвенсор – элементарный процессор – содержит 4 команды, которые кодируются двумя разрядами. 4) Устройство управления логическим ассоциативным мультипроцессором должно обеспечивать параллельное выполнение задач логического анализа. 5) Каждый секвенсор имеет ассоциативную память, а также регистры для хранения результатов логических вычислений и связи с другими секвенсорами.6) Компилятор для языка описания аппаратуры или программирования есть внешняя программа по отношению к мультипроцессору, которая обеспечивает квазиоптимальное планирование вычислительного процесса во времени и в пространстве секвенсоров с учетом ограничений на размерность блоков ассоциативной памяти. 7) Память прямого доступа обслуживает мультипроцессор и хранит программу вычислительного процесса, полученную от компилятора, для решения логической задачи. 8) Гибкая инфраструктура ассоциативной памяти обеспечивает размещение таблиц произвольной размерности. 9) GUI (Guide User Interface) предназначен для эффективного и дружественного общения с пользователем в процессе решения логических задач. 10) Точный и экономичный по времени подсчета критерий качества получаемого решения.

В последнем случае речь идет о качестве взаимодействия запроса (входного многозначного, в частности, троичного вектора m) с системой ассоциативных векторов (ассоциаторов), в результате которого должен быть сгенерирован конструктивный ответ в виде одного или нескольких ассоциаторов (A), а также численной характеристики степени принадлежности (функции качества) входного вектора m к найденному решению: $\mu(m \in A)$. Входной вектор $m = (m_1, m_2, ..., m_i, ..., m_q),\ m_i \in \{0,1,x\}$ и матрица A_i ассоциаторов

$$A_{ijr} (\in A_{ij} \in A_i \in A) = \{0,1,x\}$$

должны иметь одинаковую размерность, равную q. Далее, для удобства изложения материала, степень принадлежности m-вектора к одному ассоциатору или A-вектору будет обозначаться в виде $\mu(m \in A)$.

Существует всего 5 видов или результатов логического (теоретико-множественного) Δ-взаимодействия (пересечения) двух векторов $m \cap A$, определенных на рис. 1. Они формируют все первичные примитивные варианты реакции обобщенной ПРП-системы (Поиска, Распознавания и Принятия решения) на входное воздействие-запрос. В технологической отрасли знаний – технической диагностике (Design & Test) – указанная последовательность действий трансформируется к маршруту: поиск дефектов, их (распознавание) идентификация, (принятие решения на) восстановление работоспособности. Все три стадии технологического маршрута нуждаются в метрике оценивания решений для выбора оптимального варианта.

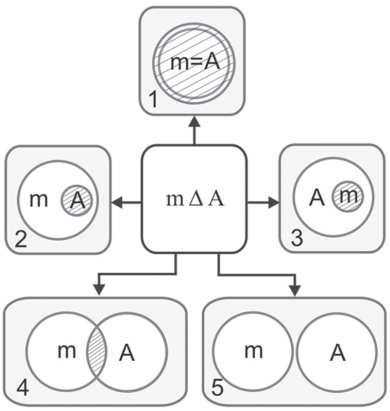

Рис. 1. Результаты пересечения двух векторов

Определение. Интегральная теоретико-множественная метрика для оценивания качества запроса есть функция качества взаимодействия многозначных векторов $m \cap A$, которая определяется средней суммой трех нормированных параметров: кодовое расстояние $d(m,A)$, функция принадлежности $\mu(m \in A)$ и эффективность использования входного запроса – функция принадлежности $\mu(A \in m)$:

$$\begin{gathered} Q = \frac{1}{3}[d(m,A) + \mu(m \in A) + \mu(A \in m)], \\ d(m,A) = \frac{1}{n}[n - card(m_i \bigcap_{i=1}^{n} A_i = \varnothing)]; \\ \mu(m \in A) = 2^{card(m \cap A) - card(A)} \leftarrow card(m \cap A) = \\ = card(m_i \bigcap_{i=1}^{n} A_i = x)\ \&\ card(A) = card(\bigcup_{i=1}^{n} A_i = x); \\ \mu(A \in m) = 2^{card(m \cap A) - card(m)} \leftarrow card(m \cap A) = \\ = card(m_i \bigcap_{i=1}^{n} A_i = x)\ \&\ card(m) = card(\bigcup_{i=1}^{n} m_i = x). \end{gathered} \quad (1)$$

Пояснения. Нормирование параметров позволяет оценивать уровень взаимодействия векторов в 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 мощность пространства вектора m ($m\cap A=m$) равна половине пространства вектора A, то функции принадлежности и качества соответственно равны:

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

$$Q(m,A)=\frac{5}{2\times 3}=\frac{5}{6}.$$

Аналогичное значение будет иметь параметр Q, если мощность пространства вектора A равна половине вектора m. Если мощность пространства пересечения $card(m\cap A)$ равна половине мощностей пространств векторов A и m, то функции принадлежности имеют значения:

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

$$Q(m,A)=\frac{4}{2\times 3}=\frac{4}{6}=\frac{2}{3}.$$

Следует также заметить, что если результат пересечения двух векторов равен пустому множеству, то степень двойки от символа «пусто» равна нулю (но не единице): $2^{card(m\cap A)=\varnothing}=2^{\varnothing}=0$. Это действительно означает, что количество общих точек при пересечении двух пространств равно нулю.

3. Процесс-модель поиска, распознавания и принятия решения

Метрика качества, представленная в (1), дает возможность оценивать близость пространственных объектов друг к другу, а также взаимодействие векторных пространств. Практическим примером полезности интегрального критерия качества может служить стрельба по цели, которая иллюстрируется ранее приведенными диаграммами (см. рис. 1) взаимодействия векторов: 1) пуля попала точно в цель и поразила ее полностью; 2) мишень поражена необоснованно большим калибром пули (снаряда); 3) калибра пули недостаточно для поражения крупной цели; 4) неэффективный и неточный выстрел снарядом большого калибра; 5) пуля пролетела мимо мишени. Для решения практических задач взаимодействия $P(m,A)$ интегральный критерий качества дает точную оценку попадания или промаха, а также эффективность использования «калибра оружия».

Аналитическая модель описания и решения логических ассоциативных отношений представлена системой уравнений:

$$\begin{gathered}
P(m,A)=\max Q_i(m\overset{n}{\underset{i=1}{\Delta}}A_i);\\
Q(m,A)=(Q_1,Q_2,...,Q_i,...,Q_n);\\
A=(A_1,A_2,...,A_i,...,A_n);\\
Q(m,A_i)=\frac{1}{3}[d(m,A_i)+\mu(m\in A_i)+\mu(A_i\in m)];\\
Q(m,A_i)=[0,1]\vee(1\leftarrow m=A);\\
\Delta=\{and,or,xor,not,slc,nop\};\\
A_i=(A_{i1},A_{i2},...,A_{ij},...,A_{is});\\
A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{msq});\\
m=(m_1,m_2,...,m_r,...,m_q).
\end{gathered}\qquad(2)$$

Здесь определены предикаты (высказывания) трех уровней иерархии: 1) Системный уровень функциональности $P(m,A)$ задает не структуры данных, а аналитическую модель вычислительного процесса в виде предиката, максимизирующего интегральный критерий принадлежности в интервале $Q(m,A)=[0,0-1,0]$, на множестве введенных операций. 2) Система предикатов среднего уровня представляется в виде вершин-таблиц графа $A=(A_1,A_2,...,A_i,...,A_m)$, логически взаимодействующих между собой. 3) Предикат нижнего уровня задает упорядоченную совокупность вектор-строк ассоциативной таблицы $A_i=(A_{i1},A_{i2},...,A_{ij},...,A_{is})$, где строка $A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{msq})$ есть истинное высказывание. Предикат $A_i=(A_{i1},A_{i2},...,A_{ij},...,A_{is})=1$ задается совокупностью ассоциативных векторов, формирующих многозначную таблицу явных решений. Поскольку функционал не имеет постоянных во времени входных и выходных переменных, то данная структура отличается от последовательной машины фон Неймана, задаваемой конечными автоматами Мили и Мура. Ассоциативность или равнозначность всех переменных в векторе $A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{msq})$ создает равные условия их существования, что означает инвариантность решения задач прямой и обратной импликации в пространстве $A_i\in A$. Ассоциативный вектор A_{ij} определяет собой явное решение, где каждая переменная задается в конечном, многозначном и дискретном алфавите $A_{ijr}\in\{\alpha_1,\alpha_2,...,\alpha_i,...,\alpha_k\}=\beta$. Взаимодействие $P(m,A)$, входного вектора-запроса $m=(m_1,m_2,...,m_r,...,m_q)$ с графом $A=(A_1,A_2,...,A_i,...,A_m)$, формирует множество решений с выбором лучшего из них по максимальному критерию качества:

$$P(m,A)=\max Q_i[m\wedge(A_1\vee A_2\vee...\vee A_i\vee...\vee A_m)].$$

Конкретное взаимодействие вершин графа между собой создает функциональность $A=(A_1,A_2,...,A_i,...,A_m)$, которая может быть оформлена в следующие структуры: 1) Единственная ассоциативная таблица, содержащая все решения логической задачи в явном виде. Преимущество – максимальное быстродействие параллельного

ассоциативного поиска решения по таблице. Недостаток – максимально высокая аппаратурная сложность решения задачи. 2) Древовидная (графовая) структура бинарных отношений между предикатами, каждый из которых формирует таблицу истинности для незначительного количества (двух) переменных. Преимущество – максимально низкая аппаратурная сложность решения задачи. Недостаток – минимальное быстродействие последовательного ассоциативного поиска решения по дереву. 3) Компромиссная графовая структура логически понятных для пользователя отношений между предикатами, каждый из которых формирует таблицу истинности для логически сильно взаимосвязанных переменных. Преимущество – высокое быстродействие параллельного ассоциативного поиска решений по минимальному числу таблиц. Сравнительно невысокая аппаратурная сложность решения задачи. Недостаток – снижение быстродействия из-за последовательной логической обработки графовой структуры решений, найденных в таблицах. Разбиение одной таблицы (ассоциативной памяти) на k частей приводит к уменьшению аппаратных затрат, выраженных в компонентах (лутах) (LUT – Look Up Table) программируемой логической матрицы. Каждая ячейка памяти создается с помощью четырех лутов. Учитывая, что ассоциативную матрицу можно представить квадратом со стороной n, то суммарные аппаратные затраты для реализации памяти системы имеют функциональную зависимость от числа разбиений, которая определяется следующим выражением:

$$Z(n) = k \times \frac{1}{4} \times \left(\frac{n}{k}\right)^2 + h = \frac{n^2}{4 \times k} + h, (h = \{n, const\}. \quad (3)$$

Второе слагаемое, равное h – есть затраты на общую схему управления системой ассоциативных памятей. Платой за уменьшение аппаратуры является снижение быстродействия обработки структуры памятей или увеличение периода анализа компонентов системы. Функциональная зависимость времени анализа логического ассоциативного графа от числа вершин или разбиений памяти имеет следующий вид:

$$T(n) = \frac{4 \times k}{t_{clk}} + \frac{4}{t_{clk}} = \frac{4}{t_{clk}}(k+1), \ (t_{clk} = const). \quad (4)$$

Здесь период обработки одной ассоциативной памяти представлен циклом, содержащим 4 синхроимпульса. Число разбиений k пропорционально увеличивает количество тактов в худшем варианте последовательного соединения памятей. Второе слагаемое $\frac{4}{t_{clk}}$ задает время, необходимое для подготовки данных на входе системы, а также для их декодирования на выходе вычислительной структуры. Функциональные зависимости аппаратных затрат и времени анализа графа ассоциативных памятей от числа вершин или разбиений представлены на рис. 2.

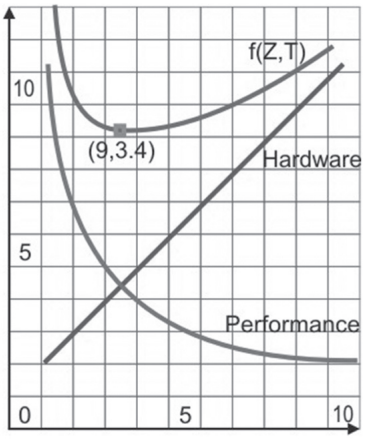

Рис. 2. Функции аппаратуры и времени от числа разбиений

Построение обобщенной функции эффективности графовой структуры от числа вершин

$$f[Z(n), T(n)] = Z(n) + T(n) = \left(\frac{n^2}{4 \times k} + h\right) + \left(\frac{4}{t_{clk}}(k+1)\right) \quad (5)$$

позволяет определить оптимальное разбиение совокупного и наперед заданного объема ассоциативной памяти. В данном случае это есть минимум аддитивной функции, который определяется значением k, обращающим производную функции в нуль. В данном случае ($n = 600$, $h = 200$, $t_{clk} = 4$) оптимальное число разбиений k для матрицы памяти, размерностью 600x600, равно 4.

4. Векторно-логический критерий качества решения

Цель введения критерия заключается в использовании только логических операций и исключении арифметических вычислений из процедуры формирования оценки взаимодействия компонентов m и A для существенного повышения быстродействия логического анализа структур информационных данных.

Идея – оценивать решение задачи взаимодействия входного вектора m с ассоциативными таблицами мощностью единиц в векторе качества Q путем использования векторных логических операций. Арифметическая оценка качества взаимодействия может формироваться сложением без усреднения приведенных критериев принадлежности и кодового расстояния, что определяется следующими формулами:

$$Q = d[m, A_{i(j)}] + \mu[m \in A_{i(j)}] + \mu[A_{i(j)} \in m],$$
$$d(m, A_{i(j)}) = card[m \overset{n(m)}{\underset{i(j)=1}{\oplus}} A_{i(j)} = 1]; \quad (6)$$

$$\mu(m \in A_{i(j)}) = card[A_{i(j)} = 1] - card[m \overset{n(m)}{\underset{i(j)=1}{\wedge}} A_{i(j)} = 1];$$

$$\mu(A_{i(j)} \in m) = card[m = 1] - card[m \overset{n(m)}{\underset{i(j)=1}{\wedge}} A_{i(j)} = 1].$$

Первый компонент, составляющий критерий, формирует степень несовпадения n-мерных векторов – кодовое расстояние, путем выполнения операции xor, второй и третий определяют степень непринадлежности результата конъюнкции к числу единиц каждого из двух взаимодействующих векторов. Понятия принадлежности и непринадлежности являются взаимодополняющими, но в данном случае технологичнее высчитывать именно непринадлежность. Таким образом, идеальный критерий качества равен нулю, когда два вектора равны между собой. Оценка качества взаимодействия двух двоичных векторов убывает по мере роста критерия от 0 к 1. Чтобы окончательно уйти от арифметических операций при подсчете уже векторного критерия качества, необходимо выражения (6) преобразовать к виду:

$$\begin{gathered} Q = d(m,A) \vee \mu(m \in A) \vee \mu(A \in m), \\ d(m,A) = m \oplus A; \\ \mu(m \in A) = A \wedge \overline{m \wedge A}; \\ \mu(A \in m) = m \wedge \overline{m \wedge A}. \end{gathered} \qquad (7)$$

Здесь критерии представлены уже не числами, а векторами, которые оценивают взаимодействие между компонентами m, A. При этом число нулей в трех оценках есть хорошо, а единицы ухудшают качество взаимодействия. Оптимизация решения логической задачи направлена на минимизацию числа единиц и максимизацию количества нулевых координат в векторах критерия качества. Для сравнения двух оценок необходимо определять мощность единиц в каждом векторе без выполнения операций суммирования. Это можно сделать, например, с помощью регистра [4] уплотнения единиц и их сдвига влево, представленного на рис. 3. Регистр позволяет за один такт выполнить сдвиг влево и уплотнить все единичные координаты n-разрядного двоичного вектора.

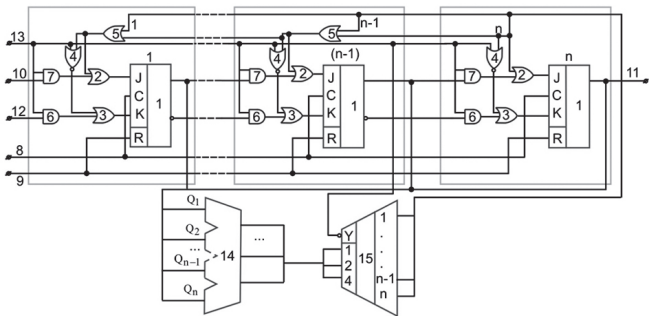

Рис. 3. Регистр уплотнения единиц

После процедуры сжатия номер последнего или правого единичного бита уплотненной серии единиц формирует индекс качества взаимодействия векторов. Для двоичных векторов $m = (1100110011\,00)$, $A = (0000111101\,01)$ определение качества их взаимодействия в соответствии с моделью (7) представлено в следующем виде (нулевые координаты отмечены точками):

m	1 1 . . 1 1 . . 1 1 . .
A	. . . . 1 1 1 1 . 1 . 1
$m \wedge A$	. . . . 1 1 . . . 1 . .
$\overline{m \wedge A}$	1 1 1 1 . . 1 1 1 . 1 1
$d(m,A) = m \oplus A$	1 1 . . . . 1 1 1 . . 1
$\mu(A \in m) = m \wedge \overline{m \wedge A}$	1 1 . . . . . 1 . . .
$\mu(m \in A) = A \wedge \overline{m \wedge A}$	. . . . . 1 1 . . . 1
$Q = d(m,A) \vee \mu(m \in A) \vee \mu(A \in m)$	1 1 . . . . 1 1 1 . . 1
$Q(m,A) = (6/12)$	1 1 1 1 1 1 . . . . .

Здесь сформирована не только оценка качества взаимодействия векторов, равная $Q(m,A) = (6/12)$, но, что самое главное, единичные координаты строки $Q = d(m,A) \vee \mu(m \in A) \vee \mu(A \in m)$ идентифицируют все те места или позиции, по которым существует некачественное взаимодействие векторов. Другой пример иллюстрирует формирование максимального критерия качества для двоичных векторов $m = (1100001100\,11)$; $A = (1100001100\,11)$, имеющих совпадение по всем координатам:

m	1 1 . . . . 1 1 . . 1 1
A	1 1 . . . . 1 1 . . 1 1
$m \wedge A$	1 1 . . . . 1 1 . . 1 1
$\overline{m \wedge A}$	. . 1 1 1 1 . . 1 1 . .
$d(m,A) = m \oplus A$	. . . . .
$\mu(A \in m) = m \wedge (\overline{m \wedge A})$	. . . . .
$\mu(m \in A) = A \wedge (\overline{m \wedge A})$	. . . . .
$Q = d(m,A) \vee \mu(m \in A) \vee \mu(A \in m)$	. . . . .
$Q = (0/12)$	. . . . .

Здесь критерий качества, равный нулю во всех 12 разрядах $Q = (0/12)$, является максимальным или самым лучшим для взаимодействующих векторов $m = (1100001100\,11)$; $A = (1100001100\,11)$, поскольку он определен минимальным числом единиц на двенадцати координатах вектора. Для сравнения двух решений, полученных в результате логического анализа, следует использовать сжатые векторы качества Q, над которыми необходимо выполнить процедуру, включающую следующие векторные операции:

$$Q(m,A) = \begin{cases} Q_1(m,A) \leftarrow Q_1(m,A) \oplus Q_1(m,A) \wedge Q_2(m,A) = 0; \\ Q_2(m,A) \leftarrow Q_1(m,A) \oplus Q_1(m,A) \wedge Q_2(m,A) \neq 0. \end{cases} \qquad (8)$$

Для двоичных векторов, представляющих собой критерии качества, выполнена процедура выбора

лучшего их них на основе выражения, представленного в (8):

$Q_1(m,A)=(6,12)$	1 1 1 1 1 1 . . . . .
$Q_2(m,A)=(8,12)$	1 1 1 1 1 1 1 1 . . . .
$Q_1(m,A)\wedge Q_2(m,A)$	1 1 1 1 1 1 . . . . .
$Q_1(m,A)\oplus Q_1(m,A)\wedge Q_2(m,A)$	. . . . .
$Q(m,A)=Q_1(m,A)$	1 1 1 1 1 1 . . . . .

Предложенная модель вычислительного процесса на основе векторных логических операций и разработанные аналитические модели и методы анализа таблиц, а также критерии качества решения позволяют использовать все упомянутое в виде инфраструктуры для поиска квазиоптимального покрытия, диагностирования одиночных и кратных дефектов программных и/или аппаратных блоков. Модель векторных вычислений может служить основой для разработки специализированной мультипроцессорной архитектуры, ориентированной на решение, например, следующих задач технической диагностики: моделирование исправного поведения цифрового устройства, заданного таблицей истинности; определение качества теста по таблице неисправностей; минимизация тестовых наборов; поиск тестовых последовательностей, распознающих дефект; идентификация кратной неисправности, имеющей место в цифровом изделии.

5. Архитектура логического ассоциативного мультипроцессора

Для анализа больших информационных объемов логических данных существует несколько практически ориентированных технологий: 1. Использование рабочей станции, где анализ решается программным путем и последовательно, поскольку существует только один процессор. Стоимость решения проблемы, а также временные затраты очень высоки. 2. Разработка специализированного параллельного процессора на основе PLD. Высокий параллелизм обработки информации компенсирует сравнительно низкую по сравнению с CPU тактовую частоту. Такое схемотехническое решение с возможностью перепрограммирования является по производительности выигрышным вариантом. Недостаток – отсутствие гибкости программных методов решения логических задач и высокая стоимость реализации системы на кристалле PLD при больших объемах промышленного выпуска изделия. 3. Лучшее решение связано с объединением достоинств CPU, PLD и ASIC. Это – гибкость программирования системы уравнений, которая позволяет оперативно корректировать спецификацию в виде исходных кодов; минимальная мощность команд и простые схемотехнические решения аппаратной реализации мультипроцессора; распараллеливание процесса решения логических задач на структуре однобитовых процессоров. Имплементация мультипроцессора в кристалл ASIC дает возможность получить максимальную тактовую частоту, минимальную стоимость чипа при больших объемах выпуска изделия, низкое энергопотребление.

Базовая конфигурация LAMP имеет сферическую структуру мультипроцессора (рис. 4), состоящую из 16 векторных секвенсоров, каждый из которых, включая граничные элементы, соединен с восемью соседними. Каждый секвенсор содержит блок векторных логических операций, ассоциативную память, блок регистров, интерфейс, а также входной и выходной мультиплексоры для связи с другими процессорами.

LAMP есть ad hoc технология и специализированное вычислительное устройство, реализуемое в кристалле ASIC, для быстрого решения логических (предикатных) уравнений ассоциативного поиска (ЛУАП) информации. Структура LAMP и модель обработки уравнений имеет прототип в виде процессора PRUS [1], разработанного доктором Stanley Hyduke (CEO Aldec, USA). Она представляет собой сеть параллельных синхронизированных векторных процессоров.

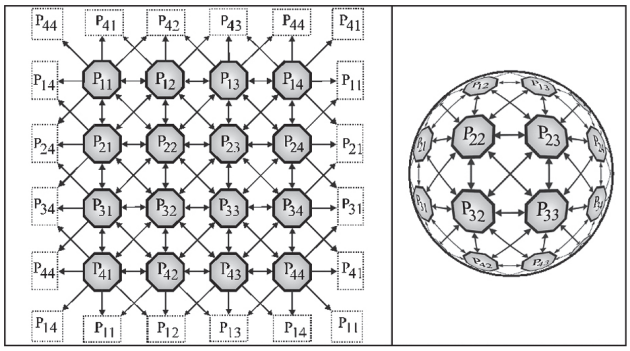

Рис. 4. Макроархитектура LAMP

Занесение информации в процессор подобно классической схеме (design flow), за исключением того, что стадия place and route заменяется фазой распределения ЛУАП между всеми логическими бит-процессорами, работающими параллельно. ЛУАП Compiler обеспечивает размещение уравнений по процессорам, задает время формирования решения на выходе каждого из них, а также планирует передачу полученных результатов другому процессору. LAMP есть эффективная сеть процессоров, которая обрабатывает систему ЛУАП и обеспечивает обмен данными между компонентами сети в процессе их решения. Простая схемотехника каждого процессора позволяет эффективно обрабатывать сверхбольшие массивы, насчитывающие миллионы бит информации, затрачивая на это в сотни раз меньше времени по сравнению с универсальным процессором. Базовая ячейка –

векторный процессор для LAMP может быть синтезирован на 200 вентилях, что дает возможность сеть, содержащую 4096 вычислителей, легко имплементировать в ASIC, используя современную силиконовую технологию. Учитывая, что затраты памяти для эмуляции ЛУАП весьма незначительны, LAMP может представлять интерес для проектирования систем управления в таких областях человеческой деятельности, как: индустрия, медицина, защита информации, геология, прогнозирование погоды, искусственный интеллект, космонавтика. LAMP представляет особый интерес для цифровой обработки данных, распознавания образов и криптоанализа. Одним из основных приложений LAMP в EDA (Electronic Design Automation) технологиях является эмуляция больших проектов, имплементируемых в ASICs и FPGA. Учитывая изложенное выше, далее формулируется проблема, цель и задачи исследования в связи с основным предназначением мультипроцессора.

Одним из возможных вариантов архитектуры мультипроцессора LAMP может служить структура, представленная на рис. 5.

Основным компонентом структуры является мультипроцессорная матрица $P=[P_{ij}],card(4\times4)$, содержащая 16 вектор-процессоров, каждый из которых предназначен для выполнения 4-х логических векторных операций над содержимым памяти данных, представленной в виде таблицы, размерностью $A=card(m\times n)$.

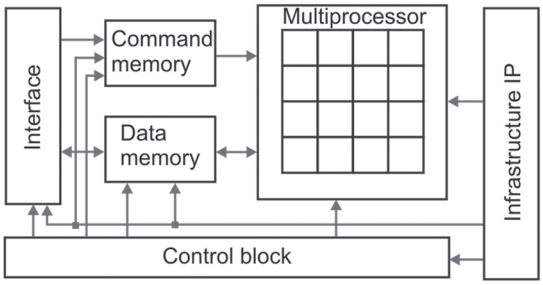

Рис. 5. Архитектура мультипроцессора LAMP

Интерфейсный блок служит для обмена данными и загрузки программы обработки данных в соответствующую память команд. Блок управления осуществляет инициализацию выполнения команд логической обработки данных и синхронизирует функционирование всех комопнентов мультипроцессора. Блок Infrastructure IP предназначен для сервисного обслуживания всех модулей, диагностирования дефектов и восстановления работоспособности компонентов и устройства в целом.

6. Инфраструктура векторно-логического анализа

Инфраструктура – совокупность моделей, методов и средств описания, анализа и синтеза структур данных для решения функциональных задач. Модель (системная) – совокупность взаимосвязанных, определенных в пространстве и времени компонентов с заданной адекватностью описывающая процесс или явление и используемая для достижения поставленной цели при наличии ограничений и метрики оценивания качества решения. Здесь ограничения есть аппаратурные затраты, время разработки и производства до появления изделия на рынке (time-to-market), подлежащие минимизации. Метрика оценивания решения при использовании модели определена двоичным вектором в дискретном булевом пространстве. Концептуальная модель вычислительного изделия в общем случае представлена совокупностью управляющего и операционного автоматов. Системная модель LAMP функциональности использует новейшую GALS (Global Asynhronus Local Synchronus) технологию создания цифровых изделий с выраженной иерархией. Модель предполагает высокое быстродействие взаимодействующих компонентов, которое обеспечивается локальной синхронизацией отдельных модулей и одновременно глобальной асинхронностью функционирования всего устройства. Если говорить о функционировании (системы) LAMP, то ее основная цель есть получение квазиоптимального решения в интегрированной задаче поиска и/или распознавания путем использования компонентов инфраструктуры, ориентированных на выполнение векторных логических операций:

$$P(m,A)=\max Q_i(m\overset{n}{\underset{i=1}{\Delta}}A_i),\qquad m=\{m_a,m_b,m_c,m_d\}. \tag{9}$$

Структура интерфейса системы, соответствующая данным функционалам, представлена на рис. 6. Все компоненты $\{A,m_a,m_b,m_c,m_d\}$ могут быть как входными, так и выходными. Двунаправленная детализация интерфейса связана с инвариантностью отношения всех переменных, векторов, А-матрицы и компонентов к входам и/или выходам инфраструктуры. Поэтому структурная модель системы LAMP может быть использована для решения любых задач прямой и обратной импликации в дискретном логическом пространстве, чем подчеркивается ее отличие от концепции автоматной модели вычислительного устройства с выраженными входами и выходами.

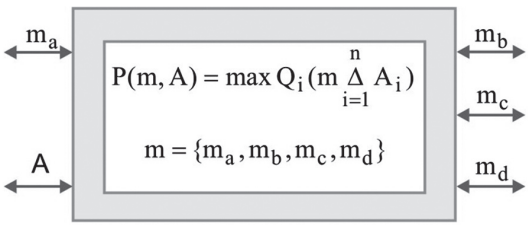

Рис. 6. Обобщенный интерфейс системы

Компоненты или регистры $m=(m_a,m_b,m_c,m_d)$ используются для получения решения в виде бу-

ферных, входных и выходных векторов, а также для идентификации оценки качества удовлетворения входного запроса. В целях детализации структуры векторного процессора или секвенсора необходимо синтезировать основные практически ориентированные процедуры анализа информационных таблиц. Процессные модели (процесс-модели), соответствующие аналитической записи вычислений, дифференцируются в две структуры: анализ А-матрицы по столбцам и по строкам. Первая из них представлена на рис. 7 и предназначена для определения множества всех допустимых решений относительно входного запроса m_b. Вторая процедурная структура (рис. 8) осуществляет поиск оптимального решения из всех возможных, найденных в первой процессной модели путем анализа строк. Кроме того, вторая структура имеет и самостоятельное применение, ориентированное на определение однозначного и многозначного решения, например, при поиске дефектов в техническом изделии.

$$m_{ai}^{m} = \bar{\vee}\,[(m_b \overset{n}{\underset{i=1}{\wedge}} A_i) \oplus m_b];$$

$$A_i = (m_b \overset{n}{\underset{i=1}{\wedge}} A_i).$$

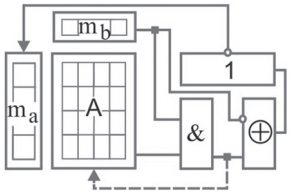

Рис. 7. Поиск всех допустимых решений

Все операции, представленные в двух моделях процессов, являются векторными. Процесс-модель анализа строк (см. рис. 7) 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 (девекторизации) дизъюнкции. Матрица A может быть модифицирована путем пересечения со входным вектором на основе использования операции $A_i = (m_b \overset{n}{\underset{i=1}{\wedge}} A_i)$, если необходимо исключить из А-таблицы все незначимые для решения координаты и векторы, отмеченные единичными значениями в m_a.

Интересное решение для задач диагностирования путем анализа строк таблицы, представленное на рис. 8, необходимо интерпретировать следующим образом.

После выполнения диагностического эксперимента формируется двоичный вектор экспериментальной проверки m_a, который маскирует А-таблицу неисправностей для поиска одиночных или кратных дефектов. Векторы m_b и m_c используются для накопления результатов выполнения операций конъюнкции и дизъюнкции. Затем осуществляется логическое вычитание 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. Схема имеет также переменную выбора режима поиска решения: single или multiple. Процесс-модель использует в качестве входного условия вектор m_a, который управляет выбором векторной операции and, or для обработки единичных $A_i(m_{ai} = 1) \in A$ или нулевых $A_i(m_{ai} = 0) \in A$ строк А-таблицы. В результате выполнения n тактов осуществляется накопление единичных и нулевых относительно значений координат вектора m_a решений в регистрах A_1, A_0 соответственно. Априори в указанные регистры заносится вектор единиц и нулей: $A_1 = 1, A_0 = 0$. После обработки всех n строк А-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. При анализе таблицы неисправностей цифрового изделия единичным координатам вектора m_b соответствуют столбцы, отождествляемые с номерами дефектов или неисправных блоков, подлежащих восстановлению или ремонту.

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

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

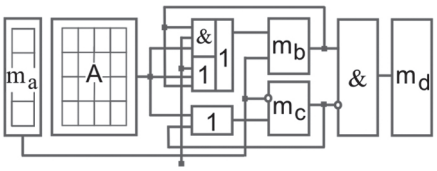

Рис. 8. Универсальная структура выбора оптимального решения

Можно пойти еще дальше в части сервисного обслуживания функциональных модулей: 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 покрыть все обнаруженные в ячейках неисправности. Технологическая и математическая культура векторной логики в данном случае предлагает простое и интересное схемотехническое решение для получения квазиоптимального покрытия, представленное на рис. 9. Преимущества: 1) Вычислительная сложность процедуры: $Z = n$ векторных операций, равное числу строк таблицы. 2) Минимум аппаратных затрат: таблица и два вектора m_b, m_a – для хранения промежуточных покрытий и накопления результата в виде единичных координат, со-

ответствующих 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. Недостаток – получение квазиоптимального покрытия, что является платой за технологичность векторной процедуры, представленной на рис. 9.

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

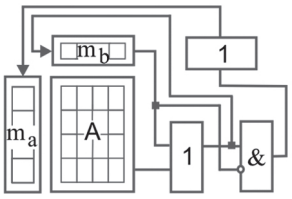

Рис. 9. Процесс-модель поиска квазиоптимального покрытия

Здесь имеется операция девекторизации, которая на последнем этапе превращает векторный результат в бит m_{ai} вектора по функции or $m_{ai} = \vee[(m_b \vee A_i) \wedge \bar{m}_b]$. В общем случае операция девекторизации в алгебре векторных операций записывается в виде <бинарная операция><вектор>: $\vee A_i, \wedge m, \bar{\wedge}(m \vee A_i)$. Обратная процедура – векторизация есть конкатенация булевых переменных: $m_a(a,b,c,d,e,f,g,h)$.

В процедуре поиска покрытия априори векторы $m_b = 0, m_a = 0$ обнуляются. Квазиоптимальное покрытие накапливается за n тактов в векторе m_a путем последовательного сдвига. Биты, заносимые в регистр m_a, формируются схемой or, которая выполняет дефекторизацию, путем анализа входного полученного результата $[(m_b \vee A_i) \wedge \bar{m}_b]$ на присутствие единиц.

Эффективность процедуры иллюстрируется поиском покрытия единицами строк всех столбцов, имеющих хотя бы одну единицу. Для матрицы покрытия, представленной в форме:

$A =$

$A_i \setminus B_j$	B_1	B_2	B_3	B_4	B_5	B_6	B_7
A_1	0	0	0	0	1	0	1
A_2	0	0	1	1	0	1	0
A_3	0	1	0	1	1	0	0
A_4	0	1	1	0	0	1	0
A_5	1	0	0	1	0	0	0

,

применение векторной процедуры поиска квазиоптимального покрытия дает следующий результат: $P = \{A_1, A_2, A_3, A_5\}$. Оптимальное покрытие для данной таблицы имеет на одну строку меньше:

$$P = \{A_1, A_4, A_5\}.$$

В целях получения более оптимального покрытия можно использовать дополнительную предварительную процедуру упорядочения покрывающей способности векторов таблицы в порядке ее убывания. Вычислительная сложность процедуры равна

$$Z = \frac{1}{2} n^2,$$

где n – число строк таблицы покрытия. Процесс-модель упорядочения векторов таблицы покрытия по убыванию количества единиц изображена на рис. 10.

$$f = \{[\vee((\overleftarrow{m}_b = A_i) \wedge (\overleftarrow{m}_A = A_r) \wedge \overleftarrow{m}_b)]_{\underset{i=\overline{1,n-1}}{\wedge}}^{r=\overline{1+1,n}}$$

$$\wedge[(m_c = A_i) \to (A_i = A_r) \to (A_r = m_c)]\}$$

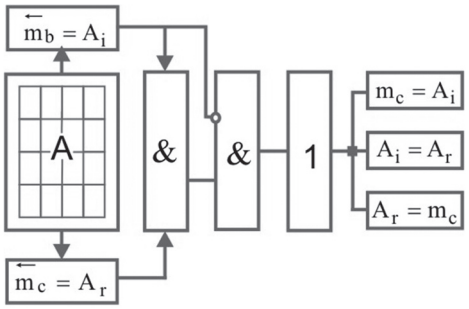

Рис. 10. Процесс-модель упорядочения векторов таблицы покрытия

Здесь левая часть логического произведения формулы есть условия обмена двумя строками, если последующая из них имеет большую покрывающую способность. В этом случае формируется единица на дизъюнктивном элементе девекторизации, которая инициирует строковый обмен. Правая часть реализует три последовательные операции обмена. Процедура упорядочения вектор-строк должна быть выполнена до запуска процесс-модели, представленной на рис. 9.

Таким образом, встроенная система диагностирования и ремонта функциональных блоков цифровой системы на кристалле имеет следующие стадии аппаратной поддержки: 1) Тестирование цифрового изделия. 2) Поиск всех допустимых решений дефектных блоков и выбор оптимального варианта. 3) Оптимизация числа ремонтных модулей для восстановления работоспособности цифрового изделия. 4) Ремонт цифровой системы на кристалле путем адресной замены неисправных компонентов.

Следующий пример интересен функциональной законченностью цикла диагностирования, когда после получения квазиоптимального покрытия данная информация используется для восстановления работоспособности дефектных ячеек памяти. Размерность модуля памяти – 13x15 ячеек не влияет на вычислительную сложность получения покрытия десяти дефектных ячеек с помощью резервных строк (2) и столбцов (5) (рис. 11).

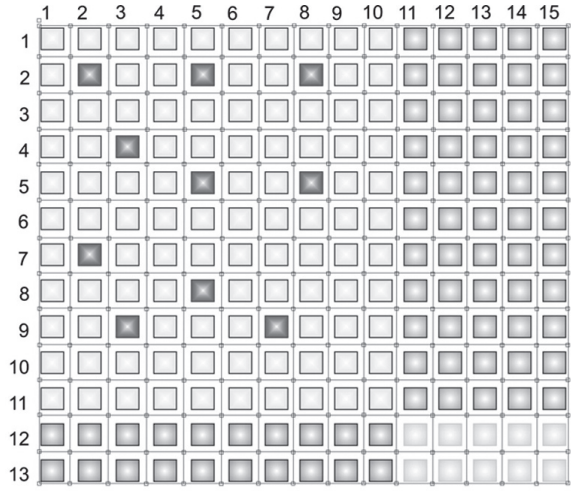

Рис. 11. Топология модуля памяти с резервом строк и столбцов

Для решения оптимизационной задачи выполняется построение таблицы покрытия неисправных ячеек, которая имеет следующий вид:

$A =$

$X_i \setminus F_{i,j}$	$F_{2,2}$	$F_{2,5}$	$F_{2,8}$	$F_{4,3}$	$F_{5,5}$	$F_{5,8}$	$F_{7,2}$	$F_{8,5}$	$F_{9,3}$	$F_{9,7}$
$C_2 \to X_1$	1						1			
$C_3 \to X_2$				1					1	
$C_5 \to X_3$		1			1			1		
$C_7 \to X_4$										1
$C_8 \to X_5$			1			1				
$R_2 \to X_6$	1	1	1							
$R_4 \to X_7$				1						
$R_5 \to X_8$					1	1				
$R_7 \to X_9$							1			
$R_8 \to X_{10}$								1		
$R_9 \to X_{11}$									1	1

Здесь столбцы соответствуют координатам дефектных ячеек, а строки идентифицируют резервные компоненты (строки и столбцы), которые могут восстановить работоспособность неисправных координат. Применение вычислительной процесс-модели, представленной на рис. 8, дает возможность получить оптимальное решение в виде вектора $m_a = \boxed{1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0}$. Данному вектору ставится в соответствие оптимальное покрытие: $R = \{X_1, X_2, X_3, X_4, X_5\} = \{C_2, C_3, C_5, C_7, C_8\}$, которое является одним из трех $R = X_1X_2X_3X_4X_5 \vee X_1X_2X_3X_5X_{11} \vee X_1X_3X_5X_7X_{11}$ возможных минимальных решений для приведенной выше таблицы неисправностей. Технологическая структурная модель встроенного диагностирования и ремонта памяти представлена на рис. 12. Она имеет четыре стадии: 1) Testing – тестирование модуля памяти (UUT – Unit Under Test) с использованием эталонной модели (MUT – Model Under Test) для формирования вектора экспериментальной проверки m_a, размерность которого соответствует числу тестовых наборов. 2) Diagnosis – поиск дефектов на основе анализа таблицы неисправностей A в соответствии с процесс-моделью, представленной на рис. 8. 3) Optimization – оптимизация покрытия дефектных ячеек ремонтными строками и столбцами на основе анализа таблицы A в соответствии с процесс-моделью, представленной на рис. 9. 4) Repairing – восстановление работоспособности модуля памяти путем замены адресов (AD – Address Decoder) неисправных строк и столбцов, представленных вектором m_a, на адреса компонентов из ремонтного запаса SM – Spare Memory.

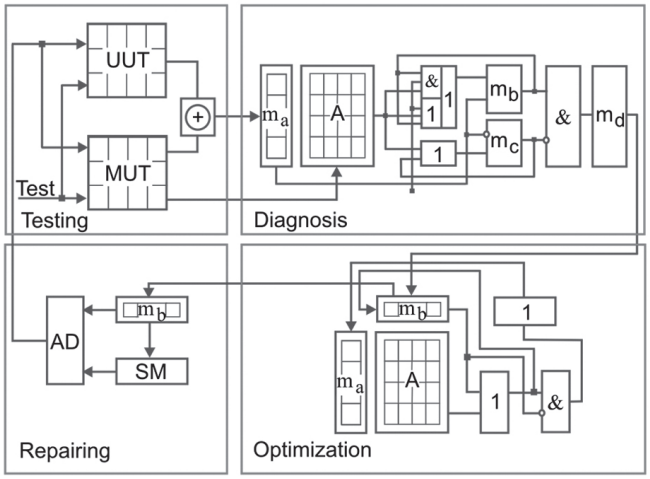

Рис. 12. Схема встроенного тестирования и восстановления памяти

Процесс-модель встроенного сервисного обслуживания, изображенная на рис. 12, работает в реальном масштабе времени и позволяет поддерживать в работоспособном состоянии, без вмешательства человека, цифровую систему на кристалле, что является интересным решением для критических технологий, связанных с дистанционной эксплуатацией изделия.

Исходя из описанных выше процесс-моделей анализа данных можно предложить относительно универсальную структуру секвенсора (рис. 13), как компонента логического ассоциативного мультипроцессора, который включает: 1) Логический процессор (LP), имеющий 5 базовых операций. 2) Ассоциативную память в виде A-матрицы для параллельного выполнения базовых операций. 4) Блок векторов m, предназначенный для параллельного обслуживания строк и столбцов A-матрицы, а также обмена данными в процессе вычислений. 5) Память прямого доступа (CM), сохраняющую команды программы обработки информации. 6) Устройство или автомат (CU), управляющий выполнением логических операций. 7) Интерфейс (I), осуществляющий связь секвенсора с другими элементами и устройствами мультипроцессора.

Логический процессор (LP) (рис. 14) осуществляет выполнение пяти операций (and, or not, xor, s1c – shift left bit crowding), которые являются базой для создания алгоритмов и процедур информационного поиска и оценивания решения. Модуль LP имеет мультиплексор на входе для выбора одного из четырех операндов, который подается на один из выбранных логических элементов, выполняющий векторную операцию. Сформированный результат через мультиплексор (элемент or) заносится в один из четырех операндов, который выбирается соответствующим адресом.

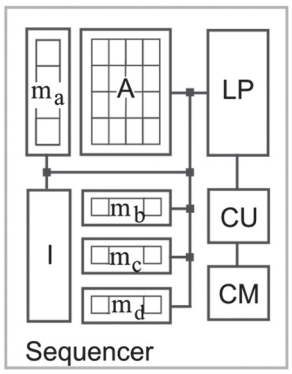

Рис. 13. Структура секвенсора

Особенности реализации логического процессора заключаются в наличии трех бинарных (and, or, xor) и двух унарных (not, slc) операций. Последние можно присоединять к такту обработки регистровых данных путем выбора одной из трех операций (not, slc, nop – нет операции). Для повышения эффективности работы логического устройства вводятся два элемента с пустой операцией. Если, например, необходимо выполнить только унарную операцию, то на уровне (слое) бинарных команд следует выбрать nop, что практически означает передачу данных через проводник (повторитель) ко второму уровню унарных 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.

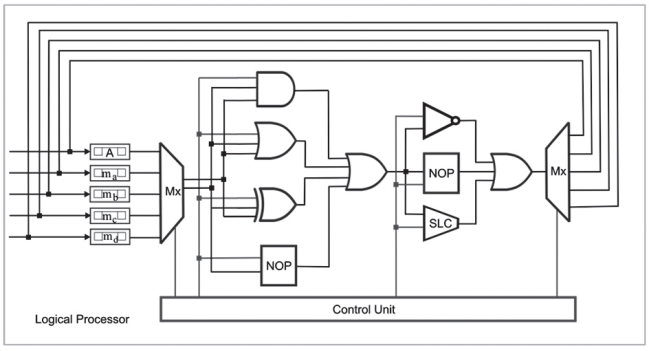

Рис. 14. Структура блока логических вычислений

В блоке логических вычислений допустимо следующее сочетание операций и операндов:

$$C = \begin{cases} \{m_a, m_b, m_c, m_d\}\Delta A_i; \\ \{m_a, m_b, m_c, m_d\}\Delta\{m_a, m_b, m_c, m_d\}; \\ \{not, nop, slc\}\{m_a, m_b, m_c, m_d, A_i\}. \end{cases}$$

$$\Delta = \{and, or, xor\}.$$

Реализация всех векторных операций блока логических вычислений для одного секвенсора в среде Verilog с последующей послесинтезной имплементацией в кристалл программируемой логики дает результаты:

Logic Block Utilization:
Number of 4 input LUTs: 400 out of 9,312 4%
Logic Distribution:
Number of occupied Slices: 200 out of 4,656 4%
Number of Slices only related logic: 200 out of 200 100 %
Total Number of 4 input LUTs: 400 out of 9,312 4%
Number of bonded IOBs: 88 out of 320 29%
Total equivalent gate count for design: 2400

Тактовая частота выполнения регистровой операции в кристалле Virtex 4, Xilinx, равна 100МГц, что на порядок выше, чем реализация аналогичных процедур на универсальном компьютере с частотой 1ГГц.

Выводы

Научная новизна представлена новыми процесс-моделями анализа табличных форм задания информации на основе использования векторных логических операций для решения задач поиска, диагностирования, распознавания образов и принятия решений в векторном дискретном булевом пространстве. Модели ориентированы на достижение высокого быстродействия процедур параллельного векторного логического анализа информации, в пределе полностью исключающего использование арифметических операций, в том числе и для подсчета критерия качества решения. Разработаны новые методы и алгоритмы решения задач диагностирования цифровых изделий, нахождения квазиоптимального покрытия, использующие векторные операции для параллельного выполнения вычислительных процессов и подсчета критериев качества.

Практическая значимость заключается в ориентации предложенных процесс-моделей, использующих векторные операции, для анализа ассоциативных таблиц на основе логического мультипроцессора с ограниченной системой команд, обеспечивающей высокое быстродействие параллельной обработки больших массивов информации, представленных в общем случае графовыми структурами ассоциативных матриц или таблиц. Дальнейшие исследования будут направлены на разработку прототипа мультипроцессора в целях решения актуальных практических задач с помощью предложенной инфраструктуры векторных логических операций.

Список литературы: 1. Zorian Y. Test Strategies for System-in-Package / Y. Zorian // Plenary Paper of IEEE East-West Design & Test Symposium (EWDTS'08). – Lviv, Ukraine. – 2008. 2. Smith L. 3D Packaging Applications, Requirements, Infrastructure and Technologies / L. Smith // Fourth Annual International Wafer-Level Packaging Conference. – San Jose, California. – September, 2007. 3. The next Step in Assembly and Packaging: System Level Integration in the package (SiP) / Editors: William Chen, W. R. Bottoms, Klaus Pressel, Juergen Wolf // SiP White Paper. International Technology Roadmap for Semiconductors. – 2007. –P. 17-23. 4. А.с. №1439682. 22.07.88. Регистр сдвига / Какурин Н.Я., Хаханов В.И., Лобода В.Г., Какурина А.Н. – 4с. 5. 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. 6. Бондаренко М.Ф. Об алгебре предикатов / М.Ф. Бондаренко, Ю.П. Шабанов–Кушнаренко // Бионика интеллекта. – Харьков: ХНУРЭ. – 2004, № 1. – С. 15–26. 7. Бондаренко М.Ф. Теория интеллекта. Учебник. / М.Ф. Бондаренко, Ю.П. Шабанов–Кушнаренко; Харьков: СМИТ. – 2006. – 592 с. 8. Бондаренко М.Ф. Модели языка / М.Ф. Бондаренко, Ю.П. Шабанов–Кушнаренко // Бионика интеллекта. – Харьков: ХНУРЭ. – 2004, № 1. – С. 27–37. 9. Акритас А. Основы компьютерной алгебры с приложениями: Пер. с англ. / А. Акритас.– М.: Мир.– 1994.– 544 с. 10. Гилл Ф. Практическая оптимизация. / Ф. Гилл, У. Мюррей, М. Райт.– М.: Мир.– 1985.– 509 с. 11. Аттетков А.В. Методы оптимизации / А.В. Аттетков, С.В. Галкин, В.С. Зарубин.– Москва: Издательство МГТУ им. Н.Э. Баумана.– 2003.– 440 с. 12. Дегтярев Ю. И. Методы оптимизации: Учебное пособие для вузов / Ю. И. Дегтярев. – М.: Сов. Радио.–1980.– 270 с. 13. Bergeron J. Writing Testbenches Using SystemVerilog / J. Bergeron // Springer Science and Business Media, Inc.– 2006. – 414 p. 14. Abramovici M. Digital System Testing and Testable Design / M. Abramovici, M.A. Breuer and A.D. Friedman.– Comp. Sc. Press.– 1998.– 652 p. 15. Densmore D. A Platform–Based taxonomy for ESL Design / Douglas Densmore, Roberto Passerone, Alberto Sangiovanni–Vincentelli // Design & Test of computers.– 2006. – P. 359–373. 16. Хаханов В.И. Проектирование и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. – Харьков: ХНУРЭ, 2009.– 484с. 17. Hahanov V.I. SIGETEST – Test generation and fault simulation for digital design / V.I. Hahanov, D.M. Gorbunov, Y.V. Miroshnichenko, O.V. Melnikova, V.I. Obrizan, E.A. Kamenuka // Proc. of Conf. «Modern SoC Design Technology based on PLD».– Kharkov.–2003.– С. 50-53. 18. Автоматизация диагностирования электронных устройств/ Ю.В.Малышенко и др./ Под ред. В.П.Чипулиса.– М.: Энергоатомиздат, 1986.– 216с. 19. 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.

Поступила в редколлегию 12.04.2010

УДК 681.326:519.713

Логічний асоціативний мультипроцесор для аналізу інформації / М.Ф. Бондаренко, В.І. Хаханов // Біоніка інтелекту: наук.-техн. журнал. – 2010. – № 2 (73). – С. 116–128.

Запропоновано нові процес-моделі аналізу табличних форм опису інформації на основі використання векторних логічних операцій для вирішення задач пошуку, діагностування, розпізнавання образів і прийняття рішень у векторному дискретному булевому просторі. Моделі зорієнтовані на досягнення високої швидкодії процедур паралельного векторного логічного аналізу інформації, що в ідеалі повністю виключає використання арифметичних операцій.

Іл. 14. Бібліогр.: 19 назв.

UDC 681.326:519.613

Logical associative multiprocessor for the information analysis. / M.F. Bondarenko, V.I. Hahanov // Bionics of Intelligence: Sci. Mag. – 2010. – № 2 (73). – С. 116–128.

Novel process-models for analyzing information in the tabular form based on using vector logical operations to solve the problems of search, diagnosis, pattern recognition and decision-making in the vector discrete Boolean space are proposed. The models are focused to realization of high-performance vector concurrent logical analysis of information that in the limit completely excludes the use of arithmetic operations.

Fig. 14. Ref.: 19 items.