УДК 658.512.011:681.326:519.713

В.И. Хаханов, Е.И. Литвинова

МУЛЬТИПРОЦЕССОР ДЛЯ АНАЛИЗА ИНФОРМАЦИОННОГО ПРОСТРАНСТВА. 1. АРХИТЕКТУРА ЛОГИЧЕСКОГО АССОЦИАТИВНОГО МУЛЬТИПРОЦЕССОРА

Предлагается архитектура мультипроцессора параллельного анализа информации, представленная в виде аналитических, графовых и табличных форм ассоциативных отношений, для поиска, распознавания и принятия решений в *n*-мерном векторном дискретном информационном пространстве.

Ключевые слова: *мультипроцессор*, *анализ информации*, *ассоциативное отношение.*

Исключение из компьютера «тяжеловесной» арифметики дает возможность трансформировать освободившиеся ресурсы в (мозгоподобную) инфраструктуру ассоциативной логики, моделирующей функциональность мозга, которая позволяет опытному человеку ежеминутно принимать правильные решения. Мозг и компьютер имеют единую технологическую основу в виде примитивных логических операций: and, or, not, xor. По мере приобретения опыта мозг и компьютер создают более сложные функциональные пространственно-временные логические преобразователи, использующие упомянутые выше примитивные операции. Специализация компьютерного изделия, ориентированная на использование только логических операций, дает возможность приблизиться к ассоциативно-логическому мышлению человека и тем самым существенно ($\times 100$) повысить быстродействие решения неарифметических задач. Исключение арифметических операций, использование параллелизма алгебры векторной логики, мультипроцессорность архитектуры создают эффективную инфраструктуру, которая объединяет математическую и технологическую культуру для решения прикладных задач. Мозгоподобность мультипроцессорной цифровой системы на кристалле есть концепция создания архитектуры и моделей вычислительных процессов для реализации свойственных мозгу неарифметических ассоциативно-логических функциональностей на современной цифровой платформе путем использования векторных логических операций и критериев в задачах поиска, распознавания и принятия решений. Рыночная привлекательность логического ассоциативного мультипроцессора (Logical Associative MultiProcessor – LAMP) определяется тысячами старых и новых логических по своей природе задач, которые в настоящее время решаются неэффективно на избыточных универсальных компьютерах с мощным арифметическим процессором. Вот некоторые актуальные для IT-рынка проблемы: 1. Анализ, синтез и коррекция синтаксических и семантических языковых конструкций (реферирование, исправление ошибок, оценивание качества текстов). 2. Распознавание видео- и аудио-образов путем их представления векторными моделями 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. 7. Целеуказание в истребителе или в автоматической системе посадки лайнера, работающей в реальном микросекундном диапазоне времени. 8. Разведение объектов во времени и в пространстве для диспетчерской службы аэропорта или оптимизация инфраструктуры городского транспорта в целях исключения коллизий. Практически все упомянутые задачи решаются в реальном масштабе времени, являются изоморфными по логической структуре процесс-моделей, использующих совокупность взаимосвязанных ассоциативных таблиц. Для их решения необходима быстродействующая и специализированная аппаратная платформа (LAMP), ориентированная на параллельное выполнение процедур поиска, распознавания и принятия решений, оцениваемых путем использования интегрального неарифметического критерия качества.

Цель исследования заключается в существенном ($\times 100$) повышении быстродействия процедур поиска, распознавания и принятия решений путем мультипроцессорной и параллельной реализации ассоциативно-логических векторных операций для анализа графовых и табличных структур данных в дискретном булевом пространстве без использования арифметических операций.

Для достижения цели необходимо решить задачи, связанные с разработкой: 1) Неарифметической метрики оценивания ассоциативно-логических решений. 2) Структуры данных и процесс-моделей решения актуальных задач. 3) Архитектуры логического ассоциативного мультипроцессора.

Сущность работы – создание инфраструктуры экспертного обслуживания запросов в реальном масштабе времени, интегрирующей мультипроцессорную систему на кристалле с аппаратом ассоциативно-логических структур данных для получения детерминированного решения, валидность (состоятельность) которого оценивается неарифметическим интегральным критерием качества взаимодействия запроса с заданным дискретным пространством.

Объект исследования – инфраструктура поиска, распознавания и принятия решений в дискретном булевом пространстве на основе использования алгебры векторной логики, мультипроцессорной платформы анализа ассоциативно-логических структур данных и неарифметического интегрального критерия качества.

Предмет исследования – ассоциативно-логические структуры данных для реализации поиска, распознавания и выбора решения на основе неарифметического интегрального критерия качества путем использования мультипроцессорной системы на кристалле, оперирующей векторными логическими операциями.

Источники. 1. Аппаратная платформа ассоциативно-логического анализа информации [1–4]. 2. Ассоциативно-логические структуры данных для решения информационных задач [5–8]. 3. Модели и методы дискретного анализа и синтеза [9–12]. 4. Мультипроцессорные средства решения информационно-логических задач [13–19]. 5. Мозгоподобные и интеллектуальные логические вычисления [20–25].

2. Интегральная метрика оценивания решения

Инфраструктура мозгоподобного мультипроцессора включает модели, методы и ассоциативно-логические структуры данных, ориентированные на аппаратную поддержку процессов поиска, распознавания и принятия решений [22–24] на основе векторных неарифметических операций. Оценка решения задачи определя-

ется векторно-логическим критерием качества взаимодействия запроса (вектора m) с системой ассоциативных векторов (ассоциаторов), в результате которого сгенерируется конструктивный ответ в виде одного или нескольких ассоциаторов, а также, пока еще 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

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

имеют одинаковую размерность, равную q. Далее степень принадлежности m-вектора к вектору A будет обозначаться как $\mu(m \in A)$.

Существует 5 типов теоретико-множественного (логического) Δ-взаимодействия двух векторов $m \cap A$, определенных на рис. 1. Они формируют все примитивные варианты реакции обобщенной ПРП-системы (поиска, распознавания и принятия решения) на входной вектор-запрос. В технологической отрасли знаний – технической диагностике (Design & Test) – указанная последовательность действий изоморфна маршруту: поиск дефектов, их распознавание, принятие решения на восстановление работоспособности. Все три стадии технологического маршрута нуждаются в метрике оценивания решений для выбора оптимального варианта.

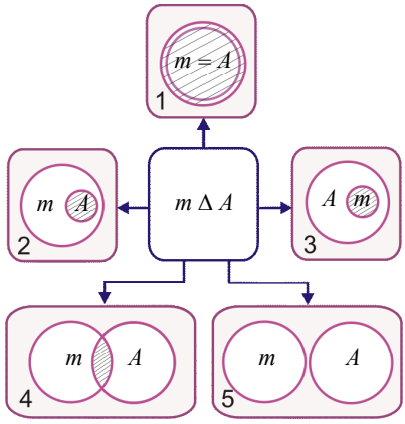

Рис. 1. Результаты пересечения двух векторов

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

$$Q = \frac{1}{3}[d(m,A) + \mu(m \in A) + \mu(A \in m)], \quad d(m,A) = \frac{1}{n}[n - card(m_i \bigcap_{i=1}^{n} A_i = \varnothing)];$$

$$\begin{aligned} \mu(m \in A) &= 2^{card(m \cap A) - card(A)} \leftarrow card(m \cap A) = \\ &= card(m_i \bigcap_{i=1}^{n} A_i = x) \,\&\, card(A) = card(\bigcup_{i=1}^{n} A_i = x); \\ \mu(A \in m) &= 2^{card(m \cap A) - card(m)} \leftarrow card(m \cap A) = \\ &= card(m_i \bigcap_{i=1}^{n} A_i = x) \,\&\, card(m) = card(\bigcup_{i=1}^{n} m_i = x). \end{aligned} \tag{1}$$

Пояснения. Нормирование параметров позволяет оценить уровень взаимодействия векторов в интервале [0,1]. Если зафиксировано предельное максимальное значение каждого параметра, равное 1, то векторы равны между собой. Мини-

мальная оценка, $Q=0$ фиксируется в случае полного несовпадения векторов по всем n координатам. Если мощность пересечения $m \cap A = m$ 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 пересечения $m \cap A = A$ равна половине пространства вектора m. Если мощность 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 заметить, что если пересечение двух векторов равно пустому множеству, то степень двойки от символа «пусто» равна нулю: $2^{card(m\cap A)=\varnothing} = 2^{\varnothing} = 0$. Это действительно означает, что количество общих точек при пересечении двух пространств равно нулю.

Цель введения векторно-логического критерия качества решения заключается в существенном повышении быстродействия при подсчете качества Q взаимодействия компонентов m и A при анализе ассоциативных структур данных путем использования только векторных логических операций. Арифметический критерий (1) без усреднения функций принадлежности и кодового расстояния можно трансформировать к виду

$$\begin{aligned}
&Q = d[m, A_{i(j)}] + \mu[m \in A_{i(j)}] + \mu[A_{i(j)} \in m],\\
&d(m, A_{i(j)}) = card\,[m \overset{n(m)}{\underset{i(j)=1}{\oplus}} A_{i(j)} = 1],\\
&\mu(m \in A_{i(j)}) = card\,[A_{i(j)} = 1] - card\,[m \overset{n(m)}{\underset{i(j)=1}{\wedge}} A_{i(j)} = 1],\\
&\mu(A_{i(j)} \in m) = card\,[m = 1] - card\,[m \overset{n(m)}{\underset{i(j)=1}{\wedge}} A_{i(j)} = 1].
\end{aligned} \tag{2}$$

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 вычислять 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 (2) преобразовать к виду

$$\begin{aligned}
&Q = d(m,A) \vee \mu(m \in A) \vee \mu(A \in m),\\
&d(m,A) = m \oplus A,\\
&\mu(m \in A) = A \wedge \overline{m \wedge A},\\
&\mu(A \in m) = m \wedge \overline{m \wedge A}.
\end{aligned} \tag{3}$$

Здесь критерии представлены уже не числами, а векторами, которые оценивают взаимодействие компонентов m, A. При этом увеличение числа нулей в трех векторах качества повышает критерий, а наличие единиц индицирует ухудшение качества взаимодействия. Для сравнения оценок необходимо определять мощность единиц в каждом векторе без выполнения операций суммирования. Это можно сделать с помощью регистра [4] (рис. 2), который позволяет за один такт выполнить сдвиг влево и уплотнить все единичные координаты *n*-разрядного двоичного вектора.

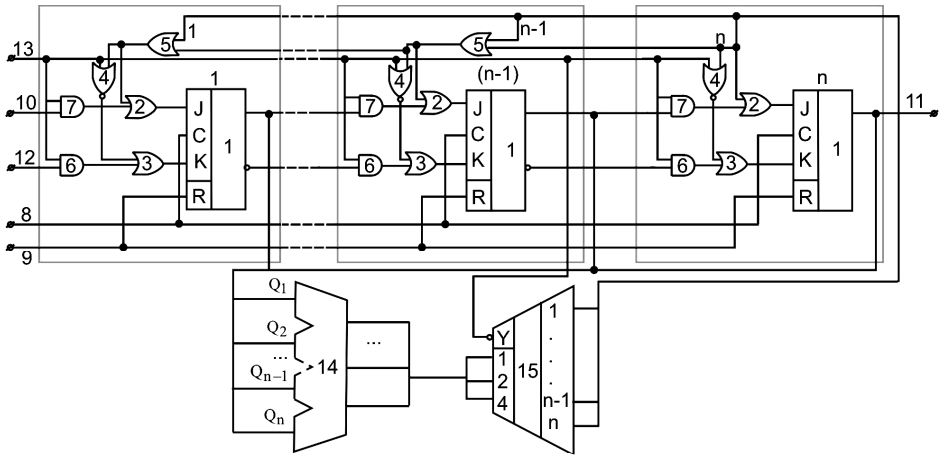

Рис. 2. Регистр сдвига и уплотнения единиц

После процедуры сжатия номер правого единичного бита уплотненной серии единиц формирует индекс качества взаимодействия векторов. Для двоичных наборов $m = (110011001100), A = (000011110101)$ определение качества их взаимодействия по формулам (3) 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 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 существенные переменные, по которым имеется некачественное взаимодействие векторов. Для сравне-

ния двух решений, полученных в результате логического анализа, используются сжатые векторы качества Q, над которыми выполняется процедура, включающая следующие векторные операции:

$$Q(m,A)=\begin{cases}Q_1(m,A)\leftarrow or[Q_1(m,A)\wedge Q_2(m,A)\oplus Q_1(m,A)]=0;\\ Q_2(m,A)\leftarrow or[Q_1(m,A)\wedge Q_2(m,A)\oplus Q_1(m,A)]=1.\end{cases} \quad (4)$$

Вектор-бит *or*-оператор девекторизации формирует двоичное битовое решение на основе применения логической операции *or* к *n* разрядам вектора существенных переменных критерия качества. Схемотехническое решение процедуры выбора

$$Q=\begin{cases}Q_1\leftarrow Y=0,\\ Q_2\leftarrow Y=1\end{cases}$$

и аналитическая процесс-модель имеют три операции, которые представлены на рис. 3.

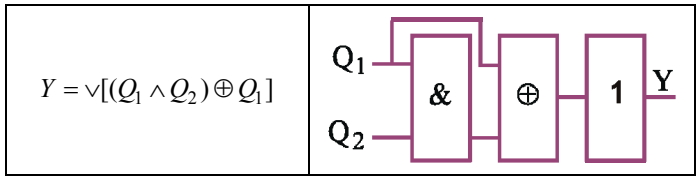

Рис. 3. Процесс-модель выбора решения

Для двоичных векторов, представляющих собой критерии качества, ниже выполнена процедура выбора лучшего их них на основании выражения, представленного в (4):

$Q_1(m,A)=(6,12)$	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 . . . . . .

Векторные логические критерии качества взаимодействия ассоциативных наборов позволяют получать оценку поиска, распознавания и принятия решения с высоким быстродействием логических параллельных операций, что особенно существенно для критических систем реального времени.

3. Процесс-модель поиска, распознавания и принятия решения

Метрика качества, представленная в (3), 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) Снаряд попал точно в цель и поразил ее полностью; 2) Мишень поражена необоснованно большим калибром снаряда; 3) Калибр снаряда недостаточен для поражения крупной цели; 4) Неэффективный и неточный выстрел снарядом большого калибра; 5) Снаряд пролетел

мимо мишени. Процесс-модель взаимодействия $P(m, A)$ сопровождается интегральным критерием качества, который оценивает не только попадание или промах, но и эффективность использования калибра оружия. Аналитическая запись обобщенной процесс-модели для выбора лучшего взаимодействия входного запроса m с системой логических ассоциативных отношений представлена в следующем виде:

$$
\begin{aligned}
&P(m, A) = \min Q_i(m \overset{n}{\underset{i=1}{\Delta}} A_i) = \vee[(Q_i \overset{j \neq i}{\underset{j=1,n}{\wedge}} Q_j) \oplus Q_i] = 0;\\
&Q(m, A) = (Q_1, Q_2, ..., Q_i, ..., Q_n);\\
&A = (A_1, A_2, ..., A_i, ..., A_n);\\
&\Delta = \{and, or, xor, not, slc, nop\};\\
&A_i = (A_{i1}, A_{i2}, ..., A_{ij}, ..., A_{is});\\
&A_{ij} = (A_{ij1}, A_{ij2}, ..., A_{ijr}, ..., A_{msq});\\
&m = (m_1, m_2, ..., m_r, ..., m_q).\\
&Q_i = d(m, A_i) \vee \mu(m \in A_i) \vee \mu(A_i \in m),\\
&d(m, A_i) = m \oplus A_i;\\
&\mu(m \in A_i) = A_i \wedge \overline{m \wedge A_i};\\
&\mu(A_i \in m) = m \wedge \overline{m \wedge A_i}.
\end{aligned}
\qquad (5)
$$

Комментарии: 1) Функциональность $P(m, A)$ задает аналитическую модель вычислительного процесса в виде высказывания, минимизирующего интегральный критерий качества. 2) Структуры данных представлены в виде вершин-таблиц графа $A = (A_1, A_2, ..., A_i, ..., A_m)$, логически взаимодействующих между собой. 3) Вершина графа задается упорядоченной совокупностью вектор-строк ассоциативной таблицы $A_i = (A_{i1}, A_{i2}, ..., A_{ij}, ..., A_{is})$ явных решений, где строка $A_{ij} = (A_{ij1}, A_{ij2}, ..., A_{ijr}, ..., A_{msq})$ представляет собой истинное высказывание. Поскольку функционал, представленный в виде таблицы, не имеет постоянных во времени входных и выходных переменных, то данная структура отличается от последовательной машины фон Неймана, задаваемой конечными автоматами Мили и Мура. Равнозначность всех переменных в векторе $A_{ij} = (A_{ij1}, A_{ij2}, ..., A_{ijr}, ..., A_{msq})$ создает одинаковые 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 минимальному критерию качества:

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

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 размещения таблицы большой размерности. 2) Древовидная (графовая) структура бинарных отношений между функциональными примитивами, каждый из которых формирует таблицу истинности для незначительного количества 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 примитивами, каждый из которых формирует таблицу истинности для логически сильно взаимосвязанных переменных. Преимущество – высокое быстродействие параллельного ассоциативного поиска решений по минимальному числу таблиц, составляющих граф, а также сравнительно невысокая аппаратурная сложность решения задачи. Недостаток – снижение быстродействия из-за последовательной логической обработки графовой структуры явных 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 [15, 16]. Каждая ячейка памяти создается с помощью четырех лутов. Учитывая, что ассоциативную матрицу можно представить квадратом со стороной n, то суммарные аппаратные затраты $Z(n)$ памяти для хранения данных и время $T(n)$ анализа логического ассоциативного графа функционально зависят от числа n разбиений таблицы или числа вершин графа:

$$Z(n) = k \times \frac{1}{4} \times \left(\frac{n}{k}\right)^2 + h = \frac{n^2}{4 \times k} + h, \ (h = \{n, \text{const}\};$$
$$T(n) = \frac{4 \times k}{t_{clk}} + \frac{4}{t_{clk}} = \frac{4}{t_{clk}}(k+1), \ (t_{clk} = \text{const}). \qquad (6)$$

Здесь h – затраты на общую схему управления системой ассоциативных памятей. Платой за уменьшение аппаратуры является снижение быстродействия обработки структуры памяти или увеличение периода анализа компонентов системы. Период обработки одной ассоциативной памяти представлен циклом из 4-х синхроимпульсов. Число разбиений k пропорционально увеличивает количество тактов в худшем варианте последовательного соединения памятей. 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. 4.

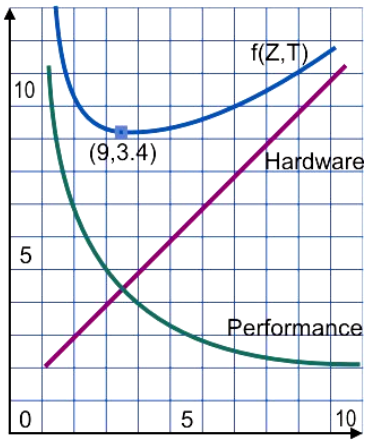

Рис. 4. Функции аппаратуры и времени от числа разбиений

Обобщенная функция эффективности графовой структуры от числа вершин

$$f[Z(n),T(n)] = Z(n) + T(n) = \left(\frac{n^2}{4\times k} + h\right) + \left(\frac{4}{t_{clk}}(k+1)\right) \qquad (7)$$

позволяет определить оптимальное разбиение совокупного и наперед заданного объема ассоциативной памяти [10 – 12]. В случае, представленном на рис. 4, лучшее разбиение есть минимум аддитивной функции, который определяется значением k, обращающим производную функции в нуль: $n \times n = 600 \times 600$, $h = 200$, $t_{clk} = 4$, $k = 4$.

Предложенная процесс-модель анализа графа ассоциативных таблиц, а также введенные критерии качества получаемых решений являются основой для разработки специализированной мультипроцессорной архитектуры, ориентированной на параллельное выполнение векторных логических операций.

4. Архитектура логического ассоциативного мультипроцессора

Для анализа больших информационных объемов логических данных существует несколько практически ориентированных технологий: 1. Использование рабочей станции для последовательного программирования задачи, где стоимость ее решения, 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. Недостатки – отсутствие гибкости, характерной программным методам 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 [15, 16]. Это – гибкость программирования, возможность корректирования исходных кодов; минимальная мощность команд и простые схемотехнические решения аппаратной реализации мультипроцессора; распараллеливание логических процедур на структуре однобитовых процессоров. Имплементация мультипроцессора в кристалл ASIC дает возможность получать максимальную тактовую частоту, минимальную стоимость чипа при больших объемах выпуска изделия, низкое энергопотребление. Объединение преимуществ перечисленных технологий определяет базовую конфигурацию LAMP, который имеет сферическую структуру мультипроцессора (рис. 5), состоящую из 16 векторных секвенсоров, каждый из которых, включая граничные элементы, соединен с восемью соседними. Структура LAMP имеет прототип в виде процессора PRUS [17], разработанного доктором Stanley Hyduke (CEO Aldec, USA).

Занесение информации в процессор подобно классической схеме (design flow), за исключением того, что стадия place and route заменяется фазой распределения программ и данных между всеми логическими бит-процессорами, работающими параллельно. Компилятор обеспечивает размещение данных 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 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 в сотни раз мень-

ше 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 хранения данных 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.

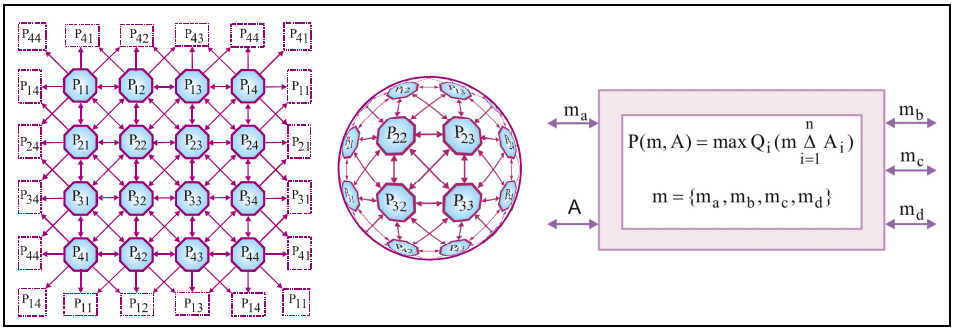

Рис. 5. Макроархитектура LAMP и интерфейс

Если говорить о функционировании (системы) LAMP, то ее основная цель – получение квазиоптимального решения в интегрированной задаче поиска и/или распознавания путем использования компонентов инфраструктуры, ориентированных на выполнение векторных логических операций:

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

Интерфейс системы, соответствующий данному функционалу, представлен на рис. 5. 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, A-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. 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.

Одним из возможных вариантов архитектуры мультипроцессора LAMP может служить структура, представленная на рис. 6. Основным компонентом структуры является мультипроцессорная матрица $P = [P_{ij}], card(4 \times 4)$, содержащая 16 вектор-процессоров, каждый из которых предназначен для выполнения пяти логических векторных операций над содержимым памяти данных, представленной в виде таблицы, размерностью $A = card(m \times n)$.

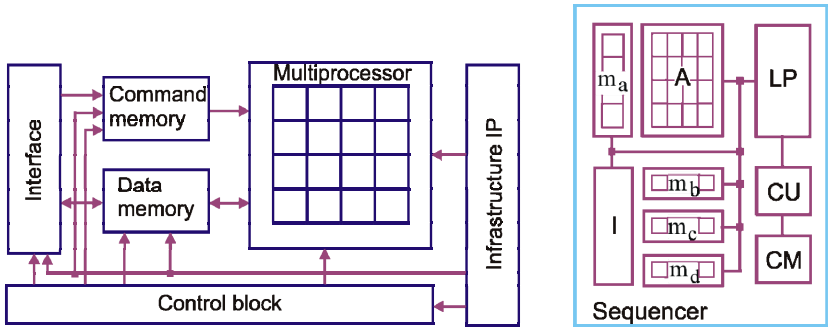

Рис. 6. Архитектура LAMP и структура секвенсора

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 компонентов мультипроцессора. Блок Infrastructure IP [1] предназначен для сервисного обслуживания всех модулей, диагностирования дефектов и восстановления работоспособности компонентов и устройства в целом. Элементарный логический ассоциативный процессор или секвенсор (см. рис. 6), входящий в состав мультипроцессора, содержит: логический процессор (LP), ассоциативную (память) *A*-матрицу для параллельного выполнения базовых операций, блок векторов *m*, предназначенный для параллельного обслуживания строк и столбцов *A*-матрицы, а также обмена данными в процессе вычислений, память прямого доступа (CM), сохраняющую команды программы обработки информации, автомат (CU) управления выполнением логических операций, интерфейс (I) связи секвенсора с другими элементами и устройствами мультипроцессора.

Логический процессор (LP) (рис. 7) 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

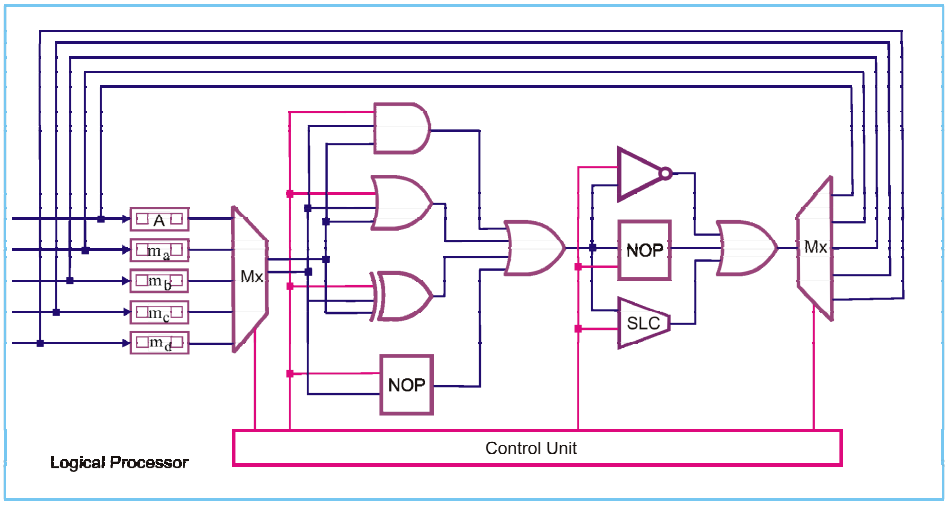

Рис. 7. Структура блока логических вычислений

LP имеет мультиплексор на входе для выбора одного из пяти операндов, который подается на выбранный логический векторный оператор. Сформированный результат через мультиплексор (элемент or) заносится в один из четырех операндов, который выбирается соответствующим адресом.

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 необходимо выполнить только унарную операцию, то на уровне бинарных команд следует выбрать nop, что практически означает передачу данных через 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. 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 следующие 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 регистровых операций в кристалле Virtex 4, Xilinx равна 100 МГц, что на порядок выше аналогичных процедур на компьютере с частотой 1ГГц.

Выводы

Научная новизна представлена процесс-моделями анализа ассоциативных таблиц на основе использования векторных логических операций для поиска, распознавания образов, принятия и оценивания решений в векторном дискретном булевом пространстве. Модели ориентированы на достижение высокого быстродействия параллельного векторного логического анализа информации, в пределе полностью исключающего использование арифметических операций, в том числе и для подсчета критерия качества решения. Предложена мультипроцессорная архитектура параллельного решения ассоциативно-логических задач с минимальным множеством векторных логических операций и полным исключением арифметических команд, что обеспечивает высокое быстродействие, минимальную стоимость и незначительное энергопотребление LAMP, имплементированного в кристалл программируемой логики.

ЛИТЕРАТУРА

1. *Zorian Y.* System-in-Package Test Strategies // Test Technology Educational Program. 2009. P. 6. URL: http://tab.computer.org/tttc/teg/ttep
2. *Smith L.* 3D Packaging Applications, Requirements, Infrastructure and Technologies // Fourth Annual International Wafer-Level Packaging Conference. San Jose, California. September, 2007. P. 4.
3. *The Next Step* in Assembly and Packaging: System Level Integration in the package (SiP) / Eds.: 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 / Н.Я. Какурин, В.И. Хаханов, В.Г. Лобода, А.Н. Какурина. 4 с.
5. *Бондаренко М.Ф., Дударь З.В., Ефимова И.А. и др.* О мозгоподобных ЭВМ // Радиоэлектроника и информатика. Харьков: ХНУРЭ. 2004. № 2. С. 89–105.
6. *Бондаренко М.Ф., Шабанов-Кушнаренко Ю.П.* Об алгебре предикатов // Бионика интеллекта. Харьков: ХНУРЭ. 2004. № 1. С. 15–26.
7. *Бондаренко М.Ф., Шабанов-Кушнаренко Ю.П.* Теория интеллекта: учебник. Харьков: СМИТ, 2006. 592 с.
8. *Бондаренко М.Ф., Шабанов-Кушнаренко Ю.П.* Модели языка // Бионика интеллекта. Харьков: ХНУРЭ, 2004. № 1. С. 27–37.
9. *Акритас А.* Основы компьютерной алгебры с приложениями: пер. с англ. М.: Мир, 1994. 544 с.
10. *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация. М.: Мир, 1985. 509 с.
11. *Аттетков А.В., Галкин С.В., Зарубин В.С.* Методы оптимизации. М.: Изд-во МГТУ им. Н.Э. Баумана, 2003. 440 с.
12. *Дегтярев Ю.И.* Методы оптимизации: учеб. пособие для вузов. М.: Сов. радио, 1980. 270 с.
13. *Bergeron J.* Writing Testbenches Using SystemVerilog // Springer Science and Business Media, Inc., 2006. 414 p.
14. *Abramovici M., Breuer M.A. Friedman, and A.D.* Digital System Testing and Testable Design. Comp. Sci. Press. 1998. 652 p.
15. *Densmore D. Passerone R., Sangiovanni-Vincentelli A.* A Platform-Based taxonomy for ESL Design // Design & Test of Computers. 2006. P. 359–373.
16. *Хаханов В.И., Литвинова Е.И., Гузь О.А.* Проектирование и тестирование цифровых систем на кристаллах. Харьков: ХНУРЭ, 2009. 484 с.
17. *Hahanov V.I., Gorbunov D.M., Miroshnichenko Y.V., et al.* SIGETEST – Test generation and fault simulation for digital design // Proc. Conf. «Modern SoC Design Technology based on PLD». Kharkov, 2003. С. 50–53.
18. *Малышенко Ю.В. и др.* Автоматизация диагностирования электронных устройств / под ред. В.П. Чипулиса. М.: Энергоатомиздат, 1986. 216 с.
19. *Хаханов В.И., Хаханова И.В., Литвинова Е.И., Гузь О.А.* Проектирование и верификация цифровых систем на кристаллах. Харьков: Новое слово, 2010. 528 с.
20. *Cohen A.A.* Addressing architecture for Brain-like Massively Parallel Computers // Euromicro Symposium on Digital System Design (DSD'04). 2004. P. 594–597.
21. *Липаев В.В.* Программная инженерия. Методологические основы: учебник. М.: Теис, 2006. 608 с.
22. *Трахтенгерц Э.А.* Компьютерные методы реализации экономических и информационных управленческих решений. СИНТЕГ, 2009. 396 с.
23. *Кузнецов О.П.* О моделировании быстрых интеллектуальных процессов обыденного мышления // Интеллектуальные системы. 1997. Т. 2. Вып. 1–4. С. 55–71.
24. *Кузнецов О.П.* Интеллектуализация поддержки управляющих решений и создание интеллектуальных систем // Проблемы управления. 2009. № 3.1. С. 64–72.

25. *Васильев С.Н.*, *Жерлов А.К.*, *Федосов Е.А.*, *Федунов Б.Е.* Интеллектуальное управление динамическими системами. М.: Физ.-мат. литература, 2000. 352 с.

Хаханов Владимир Иванович
Литвинова Евгения Ивановна
Харьковский национальный университет радиоэлектроники
(г. Харьков, Украина)
E-mail: hahanov@kture.kharkov.ua; kiu@kture.kharkov.ua Поступила в редакцию 14 сентября 2010 г.