УДК 658.512.011:681.326:519.713

М.Ф. БОНДАРЕНКО, В.И. ХАХАНОВ

МОДЕЛЬ ЛОГИЧЕСКОГО ВЫЧИСЛИТЕЛЯ ДЛЯ АНАЛИЗА АССОЦИАТИВНЫХ ТАБЛИЦ

Предлагаются быстродействующие технологии описания и решения логических задач на основе использования графов ассоциативных таблиц путем аппаратно-ориентированных параллельных средств анализа графовых и табличных структур ассоциативных отношений для получения детерминированного многозначного вывода в n-мерном дискретном пространстве. Рассматриваются проблемы создания теоретических основ логических вычислений в форме: метрики ассоциативных отношений, модели описания ассоциаций, оптимизации графовых структур данных, моделей вычислительных процессов системного уровня.

1. Актуальность создания логических вычислителей

Мотивация – создание инфраструктуры для описания, анализа и синтеза логических ассоциативных отношений, моделирующих мозгоподобные (Brain-Like – BL) вычисления. К ним можно отнести: 1. Анализ и синтез синтаксических и семантических языковых конструкций для решения многочисленных естественноязыковых задач (реферирование, исправление ошибок, анализ качества текстов). 2. Определение степени принадлежности объекта существующим ассоциативным компонентам BL-системы на основе введенных критериев качества выбора. 3. Распознавание видео- и аудио-образов путем их представления вектором существенных параметров. 4. Оперативное диагностирование неисправностей, технического состояния объекта и ремонт в процессе функционирования изделия. 5. Тестирование знаний и экспертное обслуживание объектов или субъектов для определения их валидности. Например, анализ кредитной истории для выдачи банком кредита. 6. Идентификация объекта или процесса для принятия решений в условиях неопределенности.

Существует также многообразие практически интересных для рынка задач, которые можно свести к выбору оптимального решения на основе использования логической структуры ассоциаций. 1. 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. 2. Коррекция текста в процессе его набора, когда автоматически исправляются только тривиальные ошибки, типа повторения буквы в слове, но можно корректировать более сложные ошибки, связанные с неверным окончанием, а также предлагать более приемлемые варианты порядка слов в предложении или в его части. Данная задача актуальна для 100% пользователей компьютеров. 3. Более серьезная проблема относится к критическим технологиям. Наведение на цель (выбор цели), будь то целеуказатель в истребителе, или в автоматической системе посадки лайнера. Здесь очень существенно функционирование системы наведения в реальном масштабе времени в микросекундном диапазоне измерения. 4. Обратной задачей выбора цели в критических технологиях является разведение объектов во времени и в пространстве, например, в диспетчерской службе аэропорта или оптимальной организации городского транспорта для исключения коллизий в воздушном пространстве и пробок на дорогах. Практически все упомянутые задачи должны решаться в реальном времени, они сходны по логической структуре ассоциативных многозначных векторов, которые составляют модель процесса или объекта. Они нуждаются в быстродействующей и специальной аппаратной платформе, которой может выступать логический ассоциативный мультипроцессор, ориентированный на параллельное выполнение поиска в ассоциативных структурах памяти и принятия (выбора) решения на основе использования интегрального критерия качества.

В плане предлагаемых исследований интересным представляется опубликованный в EE Times (12.2009) десяток перспективных технологий от Gartner Research Group (Gary Smith) на ближайшие годы. В нем нашли отражение только технологии, связанные с разработкой

специализированных цифровых изделий на кристаллах, хотя развитие программных технологий также будет оказывать сильное влияние на состояние рынка электроники. Глобально важными остаются технологии снижения потребляемой мощности и решения, направленные на уменьшение содержания ценных материалов в продукте. Данные технологии выступают двигателями многих направлений развития электроники, перечисленных ниже: 1. Биологическая обратная связь или электроника, управляемая мыслью. 2. Печатная электроника на основе использования органических материалов. 3. Пластиковая память на основе полимеров, проявляющих ферроэлектрические свойства. 4. Безмасочная литография на основе использовании электронного луча для создания топологии схемы. 5. Параллельная обработка данных для многоядерных гетерогенных графических процессоров. 6. Сбор энергии от механических и электрических процессов в окружающей среде. 7. Биоэлектроника и wetware, сочетающие биологические объекты и электронику для медицины. 8. Резистивное ОЗУ или мемристор с эффектом памяти как четвертый пассивный элемент электронной схемы, дополняющий резистор, конденсатор и индуктивность. 9. Переходные отверстия в кремнии (Through-Silicon-Via – TSV) для создания реальных 3D-SiP и кристаллов. 10. Различные технологии батарей на основе сочетания никеля и лития.

Цель – существенное повышение быстродействия анализа логических ассоциативных отношений путем аппаратно-ориентированной реализации параллельных средств обработки аналитических, графовых и табличных структур данных для детерминированного многозначного вывода в n-мерном дискретном пространстве.

Задачи: 1) Актуальность создания мозгоподобных вычислителей. 2) Метрика ассоциативных отношений. 3) Архитектуры ассоциаций: таблицы, графы, уравнения. 4) Оптимизация логических структур данных. 5) Модель вычислительного процесса системного уровня. 6) Практические примеры применения логического анализа ассоциативных таблиц.

Сущность – экспертное обслуживание запросов по анализу логических ассоциативных структур, ориентированных на предоставление пользователю детерминированного и многозначного вывода о принадлежности запроса к векторам n-мерного ассоциативного пространства.

Нейро-фаззи системы, изначально ориентированные на решение задач о принадлежности объекта или процесса существующим аналогам в реальном масштабе времени, остаются игрушками, если они не имплементируются в программируемый логический кристалл на основе использования структуры ассоциативных памятей. Кроме того, определенный интерес со стороны рынка электронных технологий вызывают работы, связанные с расширением функциональных возможностей структур ассоциативных памятей, использующих уже достаточно наработанную нейро-фаззи теорию. Взаимопроникновение практически ориентированных и теоретизированных подходов, положенных на современную технологическую платформу цифровых систем на кристаллах, может предоставить пользователю мощные средства для эффективного, в смысле быстродействия, решения задач искусственного интеллекта, информационно-логического синтеза и анализа словоформ и более сложных конструкций естественных языков.

Объект исследования – аппаратно-ориентированная математическая инфраструктура экспертного обслуживания задач информационно-логического синтеза и анализа в дискретном булевом пространстве на основе использования интегрального критерия качества и иерархических структур ассоциативных памятей.

Предмет исследования – логическая ассоциативная сеть экспертного обслуживания задач информационно-логического анализа и выбора решения путем использования интегрального критерия качества в дискретном булевом пространстве, ориентированная на имплементацию в иерархическую структуру ассоциативной памяти.

Источники. 1. Нейро-фаззи структуры, ориентированные на использование ассоциативной памяти [1-7]. 2. Структурные организации ассоциативных памятей для решения информационно-логических задач [8-14]. 3. Аппаратно-ориентированные методы и средства для решения информационно-логических задач [15-18].

Специализация и универсализация – две фазы спирального развития технологий компьютерной индустрии. Универсализация имеет стремление на создание еще более мощных изделий, или на предоставление пользователю новых функциональностей. Специализация

тяготеет к разработке более быстродействующих и дешевых устройств, ориентированных на эффективное обслуживание задач определенного класса. Затем разработчик начинает увеличивать функциональные мощности специального изделия, что означает новый виток на пути к универсализации.

Для решения 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.

2. Интегральная метрика определения расстояния и принадлежности

Речь идет о качестве взаимодействия запроса (входного многозначного, в частности, троичного вектора m) с системой ассоциативных векторов (ассоциаторов), результатом 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 $m = (m_1, m_2, ..., m_i, ..., m_n)$, $m_i \in \{0,1,x\}$ и матрица ассоциаторов

$$S = (A_1, A_2, ..., A_i, ..., A_k),$$

$$A_i = (a_{i1}, a_{i2}, ..., a_{ij}, ..., a_{in}),$$

$$A_i \in S,\ a_i \in \{0,1,x\}$$

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

Существует всего 5 видов или результатов взаимодействия (пересечения) двух векторов $m \cap A$, определенных на рис. 1.

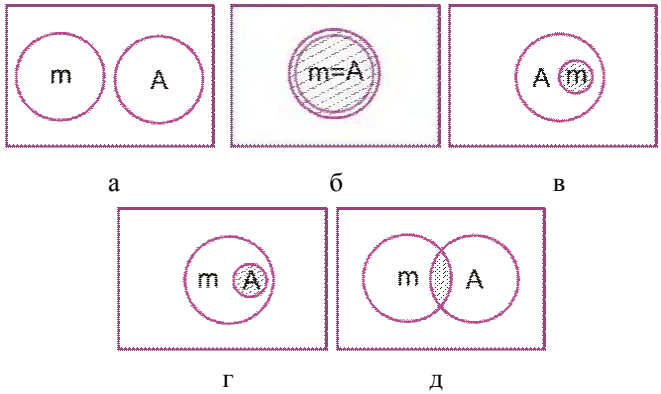

Рис. 1. Результаты пересечения двух векторов

Степень близости запроса – входного m-вектора к ассоциатору A – определяется кодовым расстоянием – мощностью множества пустых пересечений соответствующих координат взаимодействующих векторов (см. рис. 1, случай а):

$$d^*(m,A) = \mathrm{card}(m_i \bigcap_{i=1}^{n} A_i = \varnothing).$$

При кодовом расстоянии, равном 0, степень (функция) принадлежности равна 1 только в случае полного совпадения векторов (см. рис. 1, случай б). Иначе, функция принадлежности определяет часть А-векторного пространства, которая принадлежит m-вектору (см. рис. 1, случай в). Степень принадлежности здесь есть степень участия m-вектора в формировании множества состояний А-пространства при кодовом расстоянии, равном нулю. Естественно, если кодовое расстояние отлично от 0, то степень принадлежности $\mu(m \in A) = 0 \leftarrow (d^*(m,A) \succ 0)$. Два вектора, один из них входной, другой – ассоциатор или вектор-строка, определенная в троичном или многозначном алфавите, в результате пересечения образуют непустой вектор, имеющий символы X={0,1}. В данном случае функция принадлежности определяется как общее пространство, отнесенное к пространству ассоциатора, при условии, что кодовое расстояние между векторами равно 0 – пересечение двух векторов не равно пустому множеству. Для троичных векторов, имеющих символы X, формула подсчета функции принадлежности имеет вид:

$$\mu(m \in A) = \frac{2^{\mathrm{card}(m \cap A)}}{2^{\mathrm{card}(A)}} = 2^{\mathrm{card}(m \cap A) - \mathrm{card}(A)} \leftarrow$$

$$\leftarrow \mathrm{card}(m \cap A) = \mathrm{card}(m_i \bigcap_{i=1}^{n} A_i = x) \,\&\, \mathrm{card}(A) = \mathrm{card}(\bigcup_{i=1}^{n} A_i = x).$$

Примеры вычисления функций принадлежности для различных вариантов взаимодействия двух векторов представлены ниже:

$$\mu(m \in A) = \left[\frac{m = (011xxx111)}{A = (01xxxxx1x)}\right] = \frac{2^3}{2^6} = \frac{1}{8} = 0{,}125;$$

$$\mu(m \in A) = \left[\frac{m = (011000111)}{A = (011000x1x)}\right] = \frac{2^0}{2^2} = \frac{1}{4} = 0{,}25;$$

$$\mu(m \in A) = \left[\frac{m = (xxx000111)}{A = (111000xxx)}\right] = \frac{2^0}{2^3} = \frac{1}{8} = 0{,}125;$$

$$\mu(m \in A) = \left[\frac{m = (xx100011x)}{A = (xx100011x)}\right] = \frac{2^3}{2^3} = \frac{8}{8} = 1;$$

$$\mu(m \in A) = \left[\frac{m = (xxx000111)}{A = (111000111)}\right] = \frac{2^0}{2^0} = \frac{1}{1} = 1.$$

Здесь фактически рассматривается принадлежность $\mu(m \cap A \in A)$ непустого пересечения между векторами. Но в последнем примере существует неучтенная часть пространства входного m-вектора, которая не вошла в А-ассоциатор, что также иллюстрируется рассмотренным случаем - (см. рис. 1). Оценка $\mu(m \in A) = 1$, полученная в результате взаимодействия векторов, не является корректной, поскольку она не учитывает мощность векторов, участвующих в пересечении. Улучшить оценку можно путем учета введения двух функций принадлежности, а также нормированного кодового расстояния $\mu(m \in A), \mu(A \in m), d(m,A)$. Итак, кодовое расстояние – грубая оценка взаимодействия векторов. Если d*(m,A)=0, то вектор m принадлежит вектору А:

$$\begin{cases} m \in A \leftarrow d^*(m,A) = 0 \leftarrow m \cap A \neq \varnothing; \\ m \notin A \leftarrow d^*(m,A) \neq 0 \leftarrow m \cap A = \varnothing. \end{cases}$$

При выполнении первого условия необходимо вычислять степень или функцию принадлежности, учитывая, что векторы могут создавать пять видов взаимодействия (см. рис.1):

$$1)\ [m \notin A \leftarrow m \cap A = \varnothing] \rightarrow d(m, A) = \frac{1}{n}(n - d^*);$$

$$2)\ [(m \in A) \wedge (A \in m) \leftarrow m \cap A = (m \vee A) \leftarrow m = A] \rightarrow$$

$$\rightarrow \mu(m \in A) = 2^{\text{card}(m \cap A) - \text{card}(A)};$$

$$3)\ [m \in A \leftarrow m \cap A = m] \rightarrow \mu(m \in A) = 2^{\text{card}(m \cap A) - \text{card}(A)};$$

$$4)\ [A \in m \leftarrow m \cap A = A] \rightarrow \mu(A \in m) = 2^{\text{card}(m \cap A) - \text{card}(m)};$$

$$5)\ [(m \in A) \wedge (A \in m) \leftarrow m \cap A \neq (\varnothing \vee m \vee A)] \rightarrow$$

$$\rightarrow \mu(m \in A) = 2^{\text{card}(m \cap A) - \text{card}(A) - \text{card}(m)}.$$

Здесь относительно первой формулы следует подчеркнуть необходимость приведения кодового расстояния к нормированной оценке, определенной в интервале [0,1]. Данное обстоятельство вызвано тем фактом, что абсолютная целочисленная оценка не несет информации об относительном количестве несовпадений или пустых пересечений в масштабе количества (n) разрядов всего вектора. Например, два несовпадения для вектора, содержащего 20 переменных, и для вектора, имеющего 2 разряда, есть две большие разницы, которые следует учитывать при формировании функции принадлежности. Чтобы учесть все виды взаимодействия пары векторов (см. рис. 1), необходим общий и интегральный критерий качества. Все пять формул, представленных выше, необходимо объединить в интегральный критерий качества взаимодействия векторов.

Определение. Интегральная метрика для оценивания качества запроса есть функция качества (выбора) взаимодействия 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 $\mu(A \in m)$:

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

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

Ниже приведены примеры расчета интегральной функции качества для следующих 4 векторов: (00001111, 0011xx11, 0000xxxx, xxxxxxxx), которые образуют пересечения «каждый с другими» и «каждый сам с собой». Функции принадлежности и качества для всех пар представлены в таблице.

Функции качества

	$\mu(m \in A)$	$\mu(A \in m)$	$d(m, A)$	Q
m=(00001111) A=(0011xx11)	$\frac{2^{\varnothing}}{2^2} = 0$	$\frac{2^{\varnothing}}{2^0} = 0$	$\frac{8-2}{8} = 0{,}75$	0,25
m=(00001111) A=(0000xxxx)	$\frac{2^0}{2^4} = \frac{1}{16}$	$\frac{2^0}{2^0} = 1$	$\frac{8-0}{8} = 1$	0,69
m=(00001111) A=(xxxxxxxx)	$\frac{2^0}{2^8} = \frac{1}{256}$	$\frac{2^0}{2^0} = 1$	$\frac{8-0}{8} = 1$	0,67
m=(00001111) A=(00001111)	$\frac{2^0}{2^0} = 1$	$\frac{2^0}{2^0} = 1$	$\frac{8-0}{8} = 1$	1
m=(0011xx11) A=(0011xx11)	$\frac{2^2}{2^2} = 1$	$\frac{2^2}{2^2} = 1$	$\frac{8-0}{8} = 1$	1
m=(0011xx11) A=(0000xxxx)	$\frac{2^{\varnothing}}{2^4} = 0$	$\frac{2^{\varnothing}}{2^2} = 0$	$\frac{8-2}{8} = 0{,}75$	0,25
m=(0011xx11) A=(xxxxxxxx)	$\frac{2^2}{2^8} = \frac{4}{256}$	$\frac{2^2}{2^2} = 1$	$\frac{8-0}{8} = 1$	0,67
m=(0000xxxx) A=(0000xxxx)	$\frac{2^4}{2^4} = 1$	$\frac{2^4}{2^4} = 1$	$\frac{8-0}{8} = 1$	1
m=(0000xxxx) A=(xxxxxxxx)	$\frac{2^4}{2^8} = \frac{16}{256}$	$\frac{2^4}{2^4} = 1$	$\frac{8-0}{8} = 1$	0,69
m=(xxxxxxxx) A=(xxxxxxxx)	$\frac{2^8}{2^8} = 1$	$\frac{2^8}{2^8} = 1$	$\frac{8-0}{8} = 1$	1

3. Инфраструктура решения задач анализа и синтеза на логических ассоциативных графах

Инфраструктура – совокупность моделей, методов и средств представления, анализа и синтеза структур данных для решения функциональных задач.

Итак, если говорить о функционировании инфраструктуры (системы) LAMP (Logical Associative Multiprocessor), то при подаче на ее вход некоторого вектора m, маскированного двоичным вектором X, она должна сформировать на выходе вектор A, максимально непротиворечивый для запроса m, а также оценку качества полученного решения в виде функции Q:

$$\begin{cases} Q = f(m, X, A); \\ A = g(m, X, S). \end{cases}$$

Структура интерфейса системы, соответствующая данным выражениям, представлена на рис. 2.

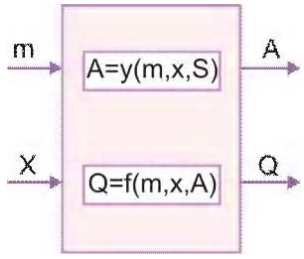

Рис. 2. Обобщенный интерфейс системы

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 пролетела мимо мишени; 2) пуля попала точно в цель и поразила ее полностью; 3) калибра пули недостаточно для поражения крупной цели; 4) мишень поражена необоснованно большим калибром пули (снаряда); 5) неэффективный и неточный выстрел снарядом большого калибра. Для решения данной практической задачи интегральный критерий качества дает точную оценку попадания (промаха), а также эффективность использования калибра оружия.

Поиск информации в Internet. Не обеспечивается глубина поиска информации по ассоциации признаков. Выдаются данные по запросу, которые чаще всего объединены функцией ИЛИ. Нет ассоциаций по минимальному кодовому расстоянию между запросом и информацией. Окончания слов оказывают большое влияние на информационный объем результирующей корзины. Здесь необходимы «умные» операции и структуры данных, а также эффективные критерии поиска информации.

Грамматическая (синтаксическая и семантическая) поддержка текстовых редакторов. Полезная функциональность – оперативная (в реальном времени) коррекция текста при его наборе в редакторе Microsoft Word. Слово с различными окончаниями (суффиксами, префиксами) воспринимается редактором как два различных слова. Решение – в плоскости анализа словоформы, когда ассоциация корня, суффикса и окончаний дают возможность исправлять ошибки при наборе текста. При этом должно учитываться синтаксическое и семантическое влияние окружающих слов. Задачи, подлежащие решению: 1) Определение оптимальной структуры логического ассоциативного графа, как формы, ориентированной на параллельное выполнение процедур анализа каждого компонента естественного языка. Структура также должна быть понятна и прозрачна для разработчика модуля IP-core, который, представляя законченную функциональность, должен компонентом входить в совокупную систему LAMP. Количество графов должно соответствовать частям речи, что представляет собой языковую структурно-логическую основу нижнего уровня. Далее над графами, как компонентами, создается интеллектуальная надстройка, определяющая практически существующие в языке ассоциативные бинарные (n-арные) логические отношения между частями речи в предложении, мощностью card = 2, 3, 4,…, n. Структура нижнего уровня предназначена для обслуживания иерархического словаря всех частей речи. Каждая из них также имеет свою иерархию в виде многозначного дерева, например, разделение всех прилагательных по первой букве. Древовидная структура обеспечивает быстрый поиск требуемого слова за счет избыточности надстройки выбора по первой букве над информативными массивами всех прилагательных. Для создания иерархического словаря LAMP, содержащего основу, окончания и признаки, необходимо иметь два компонента: электронный словарь естественного языка, к которому есть доверие со стороны разработчика, а также генератор моделей частей речи, представляющий собой анализатор (семантики и синтаксиса) естественно-языкового словаря для построения соответствующих графов (деревьев). Генератор выполняет функцию создания

полных моделей частей речи, которую можно определить как процесс интеллектуального обучения логических ассоциативных структур второго уровня. Третий уровень есть ассоциативные взаимодействия слов в предложении, где также следует разрабатывать генераторы, создающие надстройку над ассоциативными и словарными графами. Входной информацией для обучения данного уровня ассоциаций является доверительный контент (Пушкин А.С., Толстой Л.Н.), имеющий идеальные построения предложений естественного языка, которые можно принять за эталон в данном случае русского языка. Следующим может быть уровень, обслуживающий качество структуризации контента для легкости семантического понимания или восприятия совокупного сочинения. Все компоненты LAMP любого уровня иерархии в аппаратном исполнении представляют собой модули IP-cores, из которых собирается мозаика совокупной системы. Каждый модуль есть логический ассоциативный граф по форме, имплементируемый в структуру ассоциативных памятей и имеющий локальную систему управления параллельным вычислительным процессом. Для создания системы IP-cores необходимо иметь такое же количество генераторов моделей (IP-cores), которые синтезируют функциональности, как примитивные (0 и 1 уровней), так и более интеллектуальные, для верхних уровней (2-4) иерархии графа (дерева) естественного языка.

4. Модель анализа графа ассоциативных таблиц

Аналитическая модель описания и решения логических ассоциативных отношений представлена трехуровневой системой предикатов:

$$P(m,A)=Q(m,A)=[0,0;1,0];$$

$$Q(m,A)=\frac{1}{3}[d(m,A)+\mu(m\in A)+\mu(A\in m)];$$

$$P(m,A)=1\leftarrow(m=A);$$

$$P(m,A)=\max_i Q_i(m,A_i),$$

$$Q(m,A_i)=\{Q_1,Q_2,...,Q_i,...,Q_n\};$$

$$P(m,A)=m\,\Delta\,(A_1,A_2,...,A_i,...,A_k);$$

$$\Delta=\{\neg,\wedge,\vee\};$$

$$A=(A_1,A_2,...,A_i,...,A_k);$$

$$A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{ksq});\ m=(m_1,m_2,...,m_r,...,m_q); \qquad (2)$$

$$P(m,A)=m\wedge\{A_1,A_2,...,A_i,...,A_k\}=p;\ p=\{p_1,p_2,...,p_r,...,p_n\};$$

$$m\wedge(A_1\vee A_2\vee...\vee A_i\vee...\vee A_k)=\max_i Q_i(m,A_i).$$

$$p_r(\in p)=m\bigwedge_{i=\overline{1,k}}^{j=\overline{1,s}}A_{ij}\leftarrow\begin{cases}\forall i\exists j\ (m\wedge A_{ij}\neq\varnothing);\\ \exists i\,\forall j(m\wedge A_{ij}=\varnothing)\wedge d(m,A_{ij})=\max;\end{cases}$$

$$m\wedge A_{ij}=m_r\bigwedge_{r=1}^{q}A_{ijr}=\begin{cases}m\leftarrow m_r\bigwedge_{r=1}^{q}A_{ijr}=m_r;\\ A_{ij}\leftarrow m_r\bigwedge_{r=1}^{q}A_{ijr}=A_{ijr};\\ m\wedge A_{ij}\leftarrow m_r\bigwedge_{r=1}^{q}A_{ijr}=m_r\vee A_{ijr};\\ \varnothing\leftarrow\exists(m_r\bigwedge_{r=1}^{q}A_{ijr}=\varnothing);\end{cases}$$

$$P(m,A)=p_r(\in p)\leftarrow Q_i(m,p_r)=\max,\ r=\overline{1,n}.$$

Здесь определены предикаты трех уровней иерархии: 1) Системный уровень функциональности $P(m,A)=Q(m,A)$ задает не структуры данных, а аналитическую модель вычис-

лительного процесса в виде предиката, определенного интегральным критерием принадлежности в интервале $Q(m,A)=[0,0;1,0]$, на множестве введенных операций: объединение, пересечение и дополнение. 2) Система предикатов среднего уровня представляется в виде граф-упорядоченной совокупности $A=(A_1,A_2,...,A_i,...,A_k)$ взаимодействующих таблиц предикатов, принимающих только истинное значение. 3) Предикат нижнего уровня задает (упорядоченную) совокупность вектор-строк ассоциативной таблицы $A_i=(A_{i1},A_{i2},...,A_{ij},...,A_{is})$, каждая из которых $A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{ksq})$ принимает только истинное значение. 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. Иначе, ассоциативность или равнозначность всех переменных в векторе $A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{ksq})$ создает для них равные условия существования, что означает инвариантность решения задач прямой и обратной импликаций в пространстве $A_i\in A$. Ассоциативный вектор $A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{ksq})$ определяет собой явное решение, где каждая переменная задается в конечном, многозначном и дискретном алфавите $A_{ijr}\in\{\alpha_1,\alpha_2,...,\alpha_i,...,\alpha_l\}=\beta$. Взаимодействие $P(m,A)$ входного вектора-запроса $m=(m_1,m_2,...,m_r,...,m_q)$ с графом ассоциативных таблиц $A=(A_1,A_2,...,A_i,...,A_k)$ формирует множество решений $p=(p_1,p_2,...,p_r,...,p_n)$, каждое из которых имеет интегральную оценку качества $Q(m,A)=(Q_1,Q_2,...,Q_i,...,Q_n)$. Выбор лучшего из них осуществляется на основе критерия:

$$P(m,A)=p_r(\in p)\leftarrow Q_i(m,p_r)=\max,\ r=\overline{1,n}.$$

Следует заметить, что алгебраическая (аналитическая) форма представления графа – АФПГ (Graph Representation Algebraic Form – GRAF) [16]

$$m\wedge(A_1\vee A_2\vee...\vee A_i\vee...\vee A_k)=\max_i Q_i(m,A)$$

формирует точную модель взаимодействия во времени ассоциативных таблиц, ориентированных на параллельное выполнение вычислительного процесса. В данном случае она иллюстрирует факт равнозначности выполнения операции AND по отношению ко всем операндам, заключенным в скобки. Иначе, если необходима последовательность операций, представленная на рис. 3, то АФПГ будет иметь следующий вид:

$$p=mA_1\vee(mA_4\wedge mA_2)\vee mA_3\approx(A_1\vee A_4\,A_2)A_3.$$

Здесь в структуре графа можно не учитывать входную переменную m, поскольку она связана со всеми вершинами. Фактически $p=(A_1\vee A_4\,A_2)A_3$ граф определяет последовательность параллельных (конвейерных) действий, необходимых для вычисления результата – синтеза словоформы прилагательного: 1) Параллельная обработка взаимодействия входного вектора с вершинами $A_1\vee A_4$. 2) Анализ ассоциативной таблицы A_2. 3) Обработка вершины A_3 завершает процесс формирования анализа графа ассоциативных таблиц. Второй граф, представленный на рис 3, задается предикатом $p=(A_4A_2\,A_3A_1)$, который назначает процессору 4 такта для последовательной обработки вершин в целях анализа словоформы имени прилагательного.

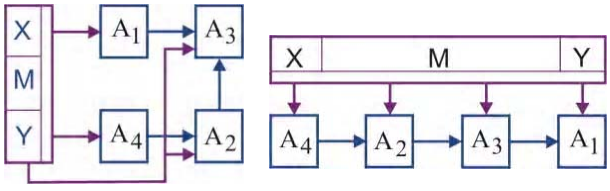

Рис. 3. Графы синтеза и анализа

Предикатная структура формирует дискретное пространство переменных или признаков, предельная мощность которого равна декартову произведению мощностей предикатов: $Card(A)=cardA_1\times cardA_2\times...\times cardA_3\times...\times cardA_k$.

Конкретное взаимодействие предикатов между собой формирует функциональность $A = (A_1, A_2, ..., A_i, ..., A_k)$, 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 за счет 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, суммарные аппаратные затраты для реализации памяти системы имеют явно выраженную функциональную зависимость от числа разбиений, которая определяется следующим выражением:

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

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:

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

где t_{clk} – период синхронизации.

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. 4.

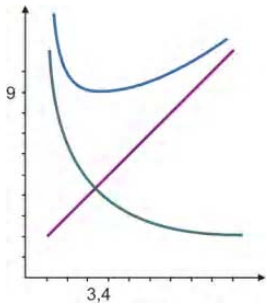

Рис. 4. Функции аппаратуры и времени от числа разбиений

Построение обобщенной функции эффективности графовой структуры от числа вершин

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

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. При этом (n=600, h=200, $t_{clk} = 4$) оптимальное число разбиений k для матрицы памяти размерностью 600x600 равно 4.

5. Диагностирование SoC на основе технологии LAMP

Логический граф ассоциативных таблиц совместно с метрикой оценивания решения предлагает интересную формулировку задачи технического диагностирования цифровых систем на кристаллах. Нельзя сказать, что данное направление является белым пятном в технической диагностике. Метод приближения [18] использует минимальное кодовое расстояние для поиска дефектов в цифровых структурах. Его недостатки связаны с ориентацией на одиночные константные неисправности булевых переменных, что порождает неприемлемо большой размер таблиц неисправностей для изделий, содержащих миллионы эквивалентных вентилей. Предлагаемый далее метод использует иерархическую инфраструктуру диагностирования, состоящую из системы таблиц неисправностей, оформленной в граф, но уже не элементарных дефектов, а неисправных функциональных компонентов (подсхем), взаимодействующих с вектором экспериментальной проверки m. Существенное дополнение метода заключается в использовании интегрального критерия качества для формирования более точного решения. Аналитическая модель процесса диагностирования представлена в виде целевой функции Z, обращающей критерий Q в минимум. Модель содержит параметры: число вершин графа G, соединенных дугами V, где для каждой вершины графа существует собственный входной вектор экспериментальной проверки из множества M:

$$Z = f(G,V,M) = \min Q,$$
$$G = (G_1, G_2, ..., G_i, ..., G_p);$$
$$V = \{V_{ij} \leftarrow (G_i G_j)\};$$
$$M = (M_1, M_2, ..., M_i, ..., M_p);$$
$$A = G_i;\ m = M_i;$$
$$A = [A] \vee [B] = [A_i] \vee [B_j],$$
$$A = (A_1, A_2, ..., A_i, ..., A_n);$$
$$B = (B_1, B_2, ..., B_j, ..., B_k).$$
$$S = \{\cup, \cap, \neg, \oplus\},$$
$$\{A_{ij}, B_{ij}, M_{ij}\} \in \{0,1,x\}.$$

В зависимости от анализа строк или столбцов матрица (таблица) $A = [A_{ij}]$ будет записываться двумя способами – по строкам или столбцам:

$$A = [A] \vee [B] = [A_i] \vee [B_j],\ A = (A_1, A_2, ..., A_i, ..., A_n),$$
$$B = (B_1, B_2, ..., B_j, ..., B_k).$$

Две формы записи служат для сокращения сложности представления формул, задающих процесс обработки матрицы, а также для улучшения понимания аналитически записанных вычислительных процессов. Входной вектор m также будет иметь различные идентификаторы $m = m^a \vee m^b$, соответствующие анализу строк или столбцов таблицы. Четыре векторные операции (and, or, not, xor) позволяют решать интересные практические задачи логического анализа на графе ассоциативных таблиц, определенных в троичном алфавите.

6. Логический метод поиска дефектов по таблице неисправностей

Рассматривается на примере транзакционного графа одного из программных модулей Row_buffer, используемых при создании IP-core вейвлет-преобразования для стандарта JPEG 2000 (рис. 5). Вершины графа представлены входными шинами и переменными, регистрами и выходными шинами. Дуги между ними означают существование транзакций между вершинами при выполнении операторов HDL-кода.

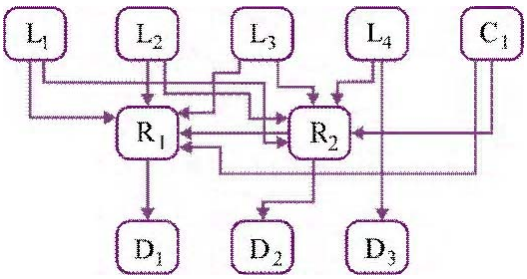

Рис. 5. Row_buffer транзакционный граф

Для данного Row_buffer графа построена таблица, задающая поведение неисправных блоков B на сгенерированном тесте A путем использования системы моделирования и генерации тестов SIGETEST [17]:

Test	B_1 L_1	B_2 L_2	B_3 L_3	B_4 L_4	B_5 C_1	B_6 R_1	B_7 R_2	B_8 D_1	B_9 D_2	B_{10} D_3	m_1^b	m_2^b
A_1	1	0	0	0	0	1	0	1	0	1	0	1
A_2	0	1	0	0	0	0	1	0	1	0	1	1
A_3	0	0	1	0	0	1	1	1	0	1	0	1
A_4	0	0	0	1	0	0	1	0	1	0	1	0
A_5	0	0	0	0	1	1	1	1	0	1	0	0
A_6	1	0	0	0	0	0	1	0	1	0	1	1
A_7	0	1	0	0	0	1	0	1	0	0	0	1
A_8	0	0	1	0	0	1	0	1	0	0	0	1
A_9	0	0	1	0	0	0	1	0	1	0	1	0
A_{10}	0	0	0	1	0	1	1	1	0	0	0	0
A_{11}	0	0	0	1	0	0	0	0	0	1	0	0
A_{12}	0	0	0	0	1	0	1	0	1	0	1	0
A_{13}	0	0	0	0	1	1	1	1	0	0	0	0

6.1. Логический метод анализа столбцов таблицы неисправностей (ТН)

Он основан на применении операции логического умножения или конъюнкции вектора экспериментальной проверки, формально рассматриваемого в качестве входного вектор-столбца или маски m, на столбцы таблицы неисправностей $m^b \wedge (B_1 \vee B_2 \vee ... \vee B_j \vee ... \vee B_m)$ и подсчете качества взаимодействия векторов $Q_j(m^b \wedge B_j)$ в целях выбора лучшего из них при наличии максимальной оценки. При этом столбец $B_j \in A$ фактически идентифицирует метрику поведения неисправности или дефектного блока на тестовых наборах. Предикатная запись процесса получения решения в виде совокупности ошибок, присутствующих в HDL-коде, представлена в следующем виде:

$$
\begin{aligned}
&P^s = P^s \bigvee_{i=1}^{n} [B_j \leftarrow \max(Q_j(m^b \bigwedge_{j=1}^{k} B_j))];\\
&P^k = P^k \bigvee_{i=1}^{n} [B_j \leftarrow (m^b \wedge B_j = B_j)] \leftarrow\\
&\leftarrow Q[m^b \wedge (\bigvee_{j=1}^{k} B_j \in P^k)] = 1 \vee \max;
\end{aligned}
\qquad (3)
$$

Здесь вектор экспериментальной проверки

$$m^b = f(A,B) \oplus f^*(A,B,L) \qquad (3)$$

есть результат проведения тестового эксперимента – сравнение функционалов (состояний выходов) эталонного $f(A,B)$ и реального $f^*(A,B,L)$ устройства с дефектами L на тестовых наборах A.

Достоинство метода – выбор всегда лучшего решения из всех возможных как для одиночных, так и для кратных дефектов. В последнем случае, если одиночный дефект не идентифицируется оценкой, равной 1, выполняется дизъюнкция таких вектор-строк (главное отличие метода от существующих технологий), которые формируют оценку качества, равную 1 или максимально близкую к единице $Q[m^b \wedge (\overset{k}{\underset{j=1}{\vee}} B_j \in P^k)] = 1 \vee \max$. По существу, в список кратных дефектов включаются такие одиночные неисправности, которые при логическом умножении на вектор экспериментальной проверки дают результат в виде соответствующего вектор-столбца. Дизъюнкция всех столбцов, составляющих решение, равна

$$\overset{k}{\underset{j=1}{\vee}} (B_j \in P^k) = m^b .$$

Используя таблицу и процедуры диагностирования (2), определяем дефектные компоненты программного кода модуля Row_buffer методом логического умножения вектор-столбцов таблицы истинности на вектор экспериментальной проверки. Здесь векторы m_1^b, m_2^b формируют результаты диагностического эксперимента, выполненные по технологии (3). Результат диагностирования одиночных и кратных дефектов имеет следующий вид:

$$\begin{aligned}
&P^s(m_1^b) = m_1^b \wedge (\overset{10}{\underset{j=1}{\vee}} B_j) = B_9 \to D_2;\\
&P^k(m_2^b) = m_2^b \wedge (\overset{10}{\underset{j=1}{\vee}} B_j) = B_1 \vee B_2 \to L_1 \vee L_2;\\
&Q(m_1^b, D_2) = 1; \qquad (4)\\
&Q[m_2^b, (L_1 \vee L_2)] = \frac{1}{3}\left(\frac{4}{13} + \frac{1}{4} + 1\right) = 0{,}52.
\end{aligned}$$

В первом случае диагноз определен в виде одного дефектного блока D_2, присутствующего в транзакционном графе, качество решения равно 1. Во втором случае процедура диагностирования выявила наличие двух дефектных модулей $L_1 \vee L_2$, которые не смогли сформировать идеальную оценку качества. Тем не менее, решение является лучшим среди всех возможных, которое максимально приближено к вектору экспериментальной проверки по критерию принадлежности $Q[m_2^b, (L_1 \vee L_2)]$.

Вычислительная сложность метода анализа столбцов определяется следующей зависимостью:

$$Z^c = 3n^2 + n^2 = 4n^2;\ Z^r = 3n + n = 4n.$$

Здесь первая оценка учитывает выполнение координатных операций над матрицей, размерностью $n \times n$. Вторая оценка определяет вычислительную сложность регистровых параллельных операций для подсчета критериев качества и обработки матрицы соответственно.

6.2. Логический метод анализа строк таблицы неисправностей

Стратегия определения ошибок программного кода по таблице неисправностей связана с анализом ее строк, состоящим из двух процедур: 1) Вычисление логического произведения конъюнкции строк, отмеченных единичными значениями вектора $A_i(m_i^b = 1)$, на отрицание дизъюнкции нулевых строк $A_i(m_i^b = 0)$ для одиночных дефектных блоков. 2) Вычисление логического произведения дизъюнкции единичных строк на отрицание дизъюнкции нулевых строк для кратных дефектных блоков:

$$P^s = (\bigwedge_{\forall m_i^b=1} A_i) \wedge (\overline{\bigvee_{\forall m_i^b=0} A_i});$$

$$P^m = (\bigvee_{\forall m_i^b=1} A_i) \wedge (\overline{\bigvee_{\forall m_i^b=0} A_i}); \quad (5)$$

Формулы интересны тем, что они не привязаны к критериям качества диагностирования, а оперируют лишь двумя компонентами, таблицей неисправностей и вектором экспериментальной проверки. Выполнение процедуры диагностирования по формулам (5) для вектора экспериментальной проверки $m^b = (0101010010010)$, заданного в последней таблице неисправностей, дает результат: $P^s(m_1^b, A) = D_2$, который не хуже, чем ранее полученный методом анализа столбцов. Для вектора экспериментальной проверки $m^b = (1110011100000)$ результат диагностирования имеет вид: $P^k(m_2^b, A) = L_1 \vee L_2$. Вычислительная сложность метода анализа строк определяется следующей зависимостью: $Z^c = n^2$; $Z^r = n$. Первая оценка предназначена для подсчета числа координатных операций, вторая определяет вычислительную сложность процесса обработки на основе регистровых параллельных операций.

7. Выводы

1. *Научная новизна* представлена алгеброй логического анализа векторных и табличных форм задания информации для решения задач поиска, диагностирования, распознавания образов и принятия решений в векторном дискретном булевом пространстве. Предложены быстродействующие модели и методы параллельного векторного логического анализа информации, в пределе полностью исключающие использование арифметических операций, в том числе и для подсчета критерия качества решения, где применяются только логические операции. В качестве примеров описаны новые методы для решения задач диагностирования и нахождения квазиоптимального покрытия, использующие векторные операции, ориентированные на распараллеливание вычислительных процессов.

2. *Практическая значимость* заключается в ориентации алгебры ассоциативных таблиц и методов их анализа на создание логического мультипроцессора с ограниченной системой команд, ориентированной на высокое быстродействие параллельной обработки больших массивов информации, представленных графовыми структурами ассоциативных матриц. Дальнейшие исследования будут направлены на разработку прототипа мультипроцессора и исследование новых практических задач с помощью предложенной алгебры.

Список литературы: 1. *Zorian Yervant.* Test Strategies for System-in-Package // The Plenary Paper of IEEE East-West Design & Test Symposium (EWDTS'08). Lvov, Ukraine. October 9-12. 2008. **2.** *Smith L.* 3D Packaging Applications, Requirements, Infrastructure and Technologies // 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.** *Какурин Н.Я., Хаханов В.И., Лобода В.Г., Какурина А.Н.* / Регистр сдвига. А.С. №1439682. 22.07.88. 4с. **5.** *Бондаренко М.Ф., Дударь З.В., Ефимова И.А., Лещинский В.А., Шабанов–Кушнаренко С.Ю.* 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. Москва. Мир. 1985. 509с. **11**. *Аттетков А.В., Галкин С.В., Зарубин В.С.* Методы оптимизации. М.: Издательство МГТУ им. Н.Э. Баумана. 2003. 440 с. **12.** *Дегтярев Ю.И.* Методы оптимизации: Учебное пособие для вузов. М.: Сов. радио, 1980. 270с. **13.** *Bergeron J.* Writing Testbenches Using SystemVerilog / J. Bergeron // Springer Science and Business Media, Inc. 2006. 414 p. **14.** *Abramovici M., Breuer M.A. and Friedman A.D.* Digital System Testing and Testable Design. Comp. Sc. Press. 1998. 652 p. **15.** *Densmore* Douglas 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., Литвинова Е.И., Гузь О.А.* Проектирование и тестирование цифровых систем на кристаллах. Харьков: ХНУРЭ. 2009. 484с. **17.** *Hahanov V.I., Gorbunov D.M., Miroshnichenko Y.V., Melnikova*

O.V., Obrizan V.I., Kamenuka E.A. SIGETEST – Test generation and fault simulation for digital design // Сб. материалов научно-практической конференции “Современные технологии проектирования систем на микросхемах программируемой логики”. Харьков. 2003. С. 50-53. **18.** *Автоматизация* диагностирования электронных устройств/ Ю.В.Малышенко и др. / Под ред. В.П. Чипулиса. М.: Энергоатомиздат, 1986. 216с.

Поступила в редколлегию 11.01.2010

Бондаренко Михаил Федорович, ректор ХНУРЭ, д-р техн. наук, профессор кафедры ПОАС ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.