РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 2 1996

ПЕРСПЕКТИВНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

УДК 681.324

В. А. Воробьев, Н. Л. Еремина

(Томск)

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ РЕКОНФИГУРАЦИИ ОТКАЗОУСТОЙЧИВОЙ ПРОЦЕССОРНОЙ МАТРИЦЫ

Предлагается программирующая система ЛОГИКА для моделирования микропрограммных клеточных автоматов (МПКА) на ЭВМ. Рассмотрена архитектура ЭВМ на основе отказоустойчивой неразрезной процессорной матрицы (ПМ). Отказоустойчивость МПКА обеспечивается реконфигурацией структуры, которая реализована программно. Обсуждается проблема диагностики ПМ и новые подходы к обеспечению отказоустойчивости.

Введение. В предлагаемом исследовании рассматривается процессорная матрица, представляющая собой прямоугольный массив процессоров, расположенный на одной пластине СБИС. Каждый процессор имеет физические координаты (i, j). Он соседствует с четырьмя процессорами с координатами $(i-1, j)$, $(i, j-1)$, $(i+1, j)$, $(i, j+1)$. С точки зрения программирования процессорная матрица представляет собой квадратную решетку размером $n \times m$. Неразрезная технология исключает замену неисправных процессорных элементов, поэтому обеспечение достаточно высокого выхода годных СБИС требует отказоустойчивости матрицы, т. е. сохранения работоспособности системы при множественных отказах элементов. Будем считать систему работоспособной в том случае, если она сохраняет исходную структуру квадратной решетки.

Исходные методы обеспечения отказоустойчивости [1, 2] состоят в следующем. Предполагается, что для каждого процессорного элемента (ПЭ) с номером (i, j) известно значение двоичного сигнала неисправности $e(i, j)$. Множество $\{e(i, j) / i = 1, ..., n, j = 1, ..., m\}$ образует синдром неисправности процессорной матрицы и служит исходной информацией для ее коррекции. В физическую структуру СБИС вводятся избыточные элементы и связи. Количество физических соседей достигает 16 элементов в зависимости от метода реконфигурации (рис. 1). В [1] избыточные элементы вводятся только по краю матрицы, в [2] они расположены в произвольных строках и столбцах исходной матрицы. Задача реконфигурации сводится к отображению логической структуры квадратной решетки в избыточную физическую структуру. Каждый процессорный элемент снабжается коммутационным окружением, состоящим из блока управления и коммутаторов (рис. 2). Коммутационное окружение процессорного элемента использует сигналы $e(i, j)$ от процессора и его соседей и обеспечивает корректирующую реконфигурацию структуры.

Существенным недостатком данного метода является требование абсолютной надежности блока управления. При этом к фатальному отказу всей матрицы ведет отказ блоков управления как исправных, так и неисправных процессоров. Это обусловлено тем, что сигнал отказа $e(i, j)$ соответствует отказу самого процессорного элемента (i, j) либо его коммутаторов. По предположению этот сигнал адекватно отображает состояние процессорного элемента и

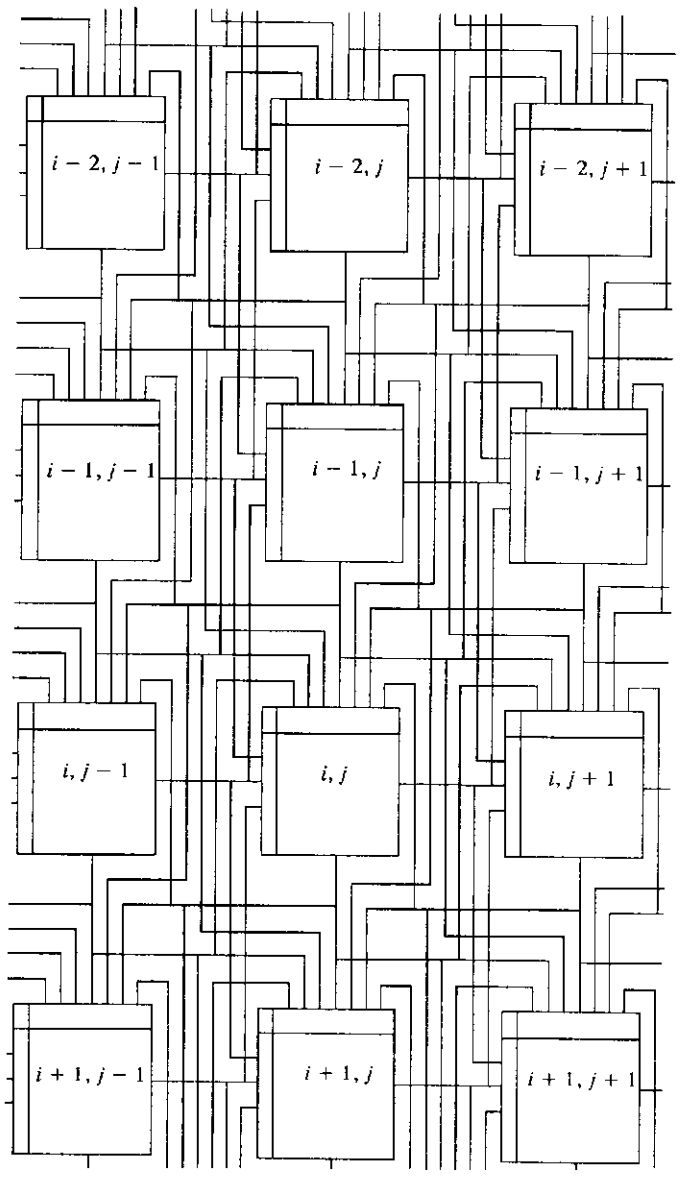

Рис. 1. Процессорная матрица (показаны только информационные связи)

адекватно обрабатывается блоками управления соседних элементов. Если же выходит из строя блок управления, то адекватность нарушается и неисправность распространяется на всю матрицу. Таким образом, желательно исключить неконтролируемый блок управления из структуры процессорной матрицы. Ниже предлагается метод программной реализации управления реконфигурацией матрицы, который позволяет снизить аппаратные затраты и существенно повысить отказоустойчивость.

Метод программной реконфигурации структуры. Множество всех блоков управления образует клеточный автомат, управляющий реконфигурацией матрицы. Он представляет собой сеть из элементарных автоматов, расположенных в узлах двумерной квадратной решетки. Каждый элементарный автомат связан по входам и выходам с некоторым числом соседей. Следует иметь в виду, что показанная на рис. 1 структура связей относится только к инфор-

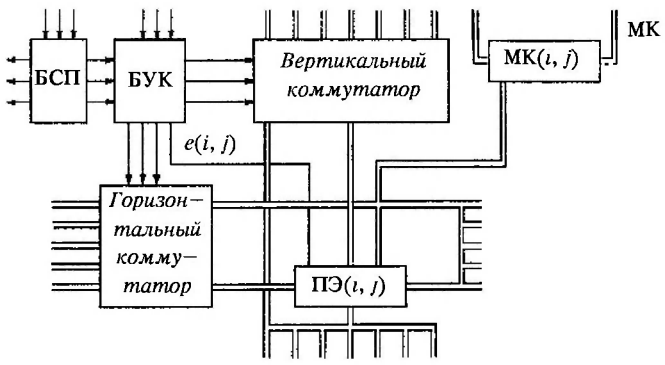

Рис. 2. Структура ячейки процессорной матрицы при схемной реализации реконфигурации

мационным шинам. Элементарные автоматы (клетки) связаны между собой одноразрядными управляющими шинами и имеют другой шаблон соседства.

Внутреннее состояние элементарного автомата определяется сигналом неисправности $e(i, j)$ и сигналами принадлежности резерву $a(i, j)$, $b(i, j)$ [2]. В простейшем случае все эти сигналы неизменны и могут рассматриваться как дополнительные входные переменные. В этом случае структура автомата представляет собой комбинационную схему без памяти и клеточный автомат уместно назвать комбинационным.

Рассмотрим подробнее структуру блока управления. Он состоит из двух частей: блока сигналов перестройки (БСП) и блока управления коммутациями (БУК). БСП элемента (i, j) воспринимает сигналы $e(i, j)$, $a(i, j)$, $b(i, j)$ от собственного процессорного элемента и сигналы $e(k, l)$, $ex(k, l)$, $ey(k, l)$ от соседних элементов. На выходе БСП вырабатываются сигналы $ex(i, j)$, $ey(i, j)$, которые называются сигналами перестройки и воздействуют на соседние элементы. БУК воспринимает сигналы перестройки $ex(i, j)$, $ey(i, j)$ от собственного БСП и БСП соседних элементов. Выходные сигналы БУК управляют коммутаторами элемента (i, j) и не воздействуют на соседей.

Присвоим всем сигналам булевы переменные. В дальнейшем сохраним наименование «сигнал» за соответствующими булевыми переменными и введем следующие обозначения: $E(i, j)$ — множество сигналов перестройки, $K(i, j)$ — множество сигналов управления коммутатором, $Z(i, j)$ — множество сигналов внутреннего состояния автомата (i, j). Пусть $d(i, j)$ обозначает множество всех соседей автомата (i, j), включая сам автомат (i, j), а $d'(i, j)$ — множество соседей, исключая автомат (i, j), т. е. $d'(i, j) = d(i, j) \setminus (i, j)$. Соответственно символы $dE(i, j)$, $dK(i, j)$, $dZ(i, j)$, $d'E(i, j)$ следует понимать как множество сигналов в указанной окрестности. Тогда описание элементарного автомата будет выглядеть следующим образом:

$$K(i, j) = F_1(dE(i, j),\ dZ(i, j)), \qquad E(i, j) = F_2(d'E(i, j),\ dZ(i, j)),$$

где F_1, F_2 — системы булевых функций.

Следует заметить, что в каждой конкретной булевой функции, например f_i из F_1, большая часть аргументов несущественна. Далее будут приведены средства описания функций, позволяющие упоминать только существенные аргументы.

Обратим внимание также на тот факт, что множество переменных $Z(i, j)$ является только входным. В общем случае некоторые элементы этого множества могут быть функциями аргументов $dE(i, j)$, $d'Z(i, j)$. Отсутствие этой зависимости означает, что блок управления (i, j) представляет собой комбинационную схему. Возникает вопрос, является ли данная схема корректной в смысле отсутствия обратных связей. Последние в принципе возможны, по-

скольку переменные множества $E(i, j)$ входят и в левую, и в правую часть системы функций:

$$E = \{E(i, j) = F_2(d' E(i, j),\ dZ(i, j)) \ / \ i = 0, ..., n - 1;\ j = 0, ..., m - 1\}.$$

Система E описывает взаимодействия на всем множестве БСП процессорной матрицы. Сопоставим с каждой переменной множества E вершину графа $G(E, V)$. Дуга v исходит из вершины e_1 и входит в вершину e_2, если e_1 — аргумент e_2 в системе E. Система корректна, если полученный граф не содержит контуров. Можно легко убедиться, что системы E из [1, 2] являются корректными в вышеуказанном смысле.

Рассмотрим систему функций

$$K = \{K(i, j) = F_1(dE(i, j),\ dZ(i, j)) \ / \ i = 0, ..., n - 1;\ j = 0, ..., m - 1\}.$$

Булевы переменные множества K служат для управления коммутаторами, причем каждая переменная y_i из K включает одну из связей между процессорными элементами. В каждом процессорном элементе должны быть включены одна вертикальная, одна горизонтальная связи и не более чем одна перемычка: вертикальная либо горизонтальная. Следовательно, для каждого элемента (i, j) не более трех элементов множества $K(i, j)$ принимают единичное значение. Множество $K(i, j)$ можно рассматривать как микрокоманду для коммутационного окружения процессора (i, j), а вся система K есть параллельная микрокоманда [3] реконфигурации процессорной матрицы.

Итак, клеточное множество, состоящее из блоков управления всех элементов матрицы, образует параллельный комбинационный микропрограммный клеточный автомат (МПКА). Сущность программной реализации структуры состоит в вычислении параллельной микрокоманды за пределами процессорной матрицы, т. е. в управляющей ЭВМ, передаче этой микрокоманды в процессорную матрицу и исполнении реконфигурации средствами коммутаторов. При этом блок управления удаляется из схемы коммутационного окружения. Доставка параллельной микрокоманды в процессорную матрицу происходит специальными средствами.

Для реализации метода программной реконфигурации структуры разработаны:

1) архитектура матричной ЭВМ, допускающей программную реализацию алгоритмов реконфигурации структуры;

2) язык ЛОГИКА1 для описания клеточных автоматов;

3) программа трансляции с языка ЛОГИКА1 в язык троичных матриц, удобный для дальнейшей обработки;

4) программа-интерпретатор, вычисляющая функции микропрограммного клеточного автомата по его описанию;

5) средства визуализации состояния процессорной матрицы на дисплее.

Архитектура матричной ЭВМ. Архитектура матричной ЭВМ, допускающей программную реализацию алгоритмов реконфигурации структуры матрицы (рис. 3), содержит мониторную подсистему, массовую память и решающее поле (РП).

Мониторная подсистема обеспечивает: а) общение системы с внешним миром (программистом, файлами данных, устройствами ввода-вывода); б) подготовку программ для решающего поля; в) тестирование РП и выдачу микрокоманды реконфигурации для РП. Естественный путь реализации мониторной подсистемы — использование персональной ЭВМ.

Массовая память в простейшем случае реализуется средствами ПЭВМ, но при высокой интенсивности обменов с РП необходим быстродействующий канал, о котором речь пойдет ниже.

Решающее поле является быстродействующей приставкой к ПЭВМ и состоит из процессорной матрицы и дополнительного оборудования.

Процессорная матрица, описанная в предыдущем разделе, является основой решающего поля. С архитектурной точки зрения имеет смысл тот случай, когда корректирующая реконфигурация структуры удалась. Полученную

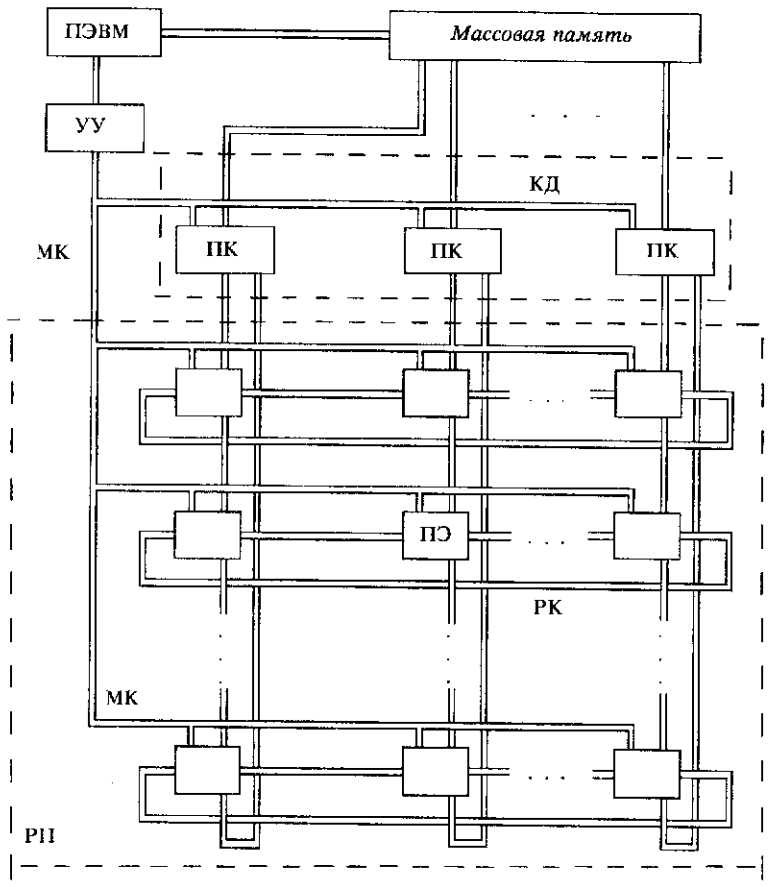

Рис. 3. Структура матричной ЭВМ

квадратную решетку исправных связей будем называть регулярным каналом (РК), обеспечивающим информационные взаимодействия между процессорными элементами. Координаты (i, j) процессорных элементов здесь соответствуют их месту в регулярном канале, а не физическому положению на пластине СБИС. Такие координаты называются логическими.

Дополнительное оборудование решающего поля зависит от способа функционирования процессорной матрицы. Если ПЭ не может хранить и выполнять программу, то строится система класса ОКМД (по Флинну [4]), т. е. решающее поле снабжается дополнительным устройством управления (УУ) с памятью микропрограмм. УУ получает микропрограмму от мониторной подсистемы и передает микрокоманды в процессорную матрицу по магистральному каналу (МК). Процессорный элемент (i, j) выполняет или не выполняет очередную микрокоманду в зависимости от состояния (i, j)-го триггера активности, а содержание сигнала этого триггера может зависеть от локальных результатов вычислений в (i, j)-м ПЭ или передается в ПЭ (i, j) из УУ по специальному каналу активности (КА). Канал активности можно реализовать как подканал МК или как специальную микрокоманду, передаваемую по МК. В этом случае по МК необходимо передавать и адресную информацию, т. е. логические координаты (i, j).

Для обеспечения потока данных в РП и обратно необходим быстродействующий канал данных (КД). В простейшем случае эту роль может выполнять МК, который становится шиной данных и команд. Однако для эффективного использования решающего поля следует обеспечить широкий канал обмена

РП с массовой памятью системы. Обычно такой КД делается дополнительно к магистральному каналу [4], но в случае отказоустойчивой матрицы на неразрезной СБИС возникает специфическая трудность: логические и физические координаты ПЭ не совпадают. Это значит, что КД должен перестраиваться вместе с перестройкой процессорной матрицы. Сложность физической структуры КД будет сравнима со сложностью физической структуры РК, некоторое представление о которой дает рис. 1. Учитывая этот факт, возложим функции КД на РК, аналогично будет устроен вывод результатов. Дополнительно к РК появится переключатель канала (ПК) с режима ввода-вывода на режим внутренних взаимодействий.

Структура матричной ЭВМ показана на рис. 3 безотносительно к способу функционирования процессорной матрицы. Все устройства на этом рисунке присутствуют и в том случае, когда каждый ПЭ может выполнять свою программу независимо от остальных, по типу МКМД. Несколько изменяются только функции МК и УУ. Рассмотрим эти особенности.

Во-первых, программа (микропрограмма), передаваемая по МК, не исполняется, а запоминается всеми или только некоторыми процессорными элементами в РП. Подмножество ПЭ, запоминающих программу, задается теми же триггерами активности.

Во-вторых, все вычислительные операции в системе типа МКМД естественно разделить на классы: локальные и глобальные. Локальными являются рутинные вычисления над данными, расположенными в памяти ПЭ. Сюда же можно отнести и обмены информацией с соседями, выполняемые параллельно на регулярном канале. Глобальные операции выполняются над общими для всех ПЭ данными (например, активизация ПЭ, рассылка одного значения всем ПЭ, ввод и вывод). К ним относятся: переходы (ветвления) по условиям, общим для всех ПЭ, синхронизации вычислений, прерывания, запуски и остановы РП. Все эти операции эффективнее исполнять с помощью УУ и МК, общих для всей матрицы. Таким образом, в МКМД-системе магистральный канал не только передаст команды и данные в РП, но и выполняет некоторые команды.

Задача данного раздела — указать те архитектурные особенности, которые вызваны отказоустойчивостью РП, т. е. средства получения синдрома отказов и доставки микрокоманды перестройки в РП. Желательно, чтобы эти средства были выбраны из уже перечисленных и не усложняли их. Таковыми могут быть каналы РК и МК при следующих предпосылках:

1. Наличие команды МК, устанавливающей РП в исходное состояние. В этом состоянии логические координаты ПЭ совпадают с физическими координатами, а информационные связи (коммутации) между ПЭ отсутствуют.

2. Наличие команд МК, устанавливающих коммутацию для любого элемента РП. С помощью этих команд может быть сформирована любая конфигурация регулярного канала.

3. Достаточная надежность магистрального канала, обеспечиваемая простотой структуры МК, избыточностью и/или возможностью пережигания мешающих соединений (например, коротких замыканий) еще в процессе изготовления матрицы. В любом случае обеспечение надежности МК проще, чем обеспечение отказоустойчивости всей матрицы.

Стратегия диагностики и коррекции структуры процессорной матрицы состоит в следующем:

1. Матрица приводится в исходное состояние, и в этом состоянии каждый ПЭ получает по МК часть тестирующей последовательности команд. В ОКМД-системе частичный тест исполняется синхронно, в МКМД — асинхронно.

2. По окончании частичного теста по МК передается коммутация и получается некоторый вариант РК. В этом состоянии происходит сравнение результатов частичного теста ПЭ с его соседями. Если ПЭ «не согласен» со своим соседом, то этот факт фиксируется как в ПЭ (i, j), так и в мониторной подсистеме.

3. П. 2 повторяется с различными вариантами структуры РК до тех пор, пока все ПЭ (i, j) не сравнятся со всеми своими физическими соседями. По-

скольку сравнение происходит параллельно, то эта процедура достаточно эффективна.

4. По окончании п. 3 происходит переход на п. 1 для ввода и реализации очередного частичного теста. Вся процедура из пп. 1—4 повторяется до исчерпания тестовой последовательности команд. В результате получается синдром несогласия, в котором для каждого ПЭ (i, j) составлен список соседей, с которыми он «не согласен».

5. Синдром несогласия преобразуется в синдром неисправности процессорной матрицы. Эта процедура может быть задана микропрограммным клеточным автоматом и реализована программно в мониторной подсистеме точно так же, как и МПКА реконфигурации.

6. В мониторной подсистеме на основе синдрома неисправности вычисляется микрокоманда конфигурации для каждого ПЭ (i, j) и его логические координаты (i', j').

7. По МК в РП передаются микрокоманды коммутации для каждого ПЭ (i, j).

8. По МК в РП передаются логические координаты каждого ПЭ. Процессорная матрица готова к работе.

Нетрудно видеть, что предлагаемая стратегия диагностики и коррекции привязана к тем методам реконфигурации, которые развиты в [1, 2]. Синдром несогласия содержит информацию о неисправных ПЭ и информационных связях или ключах коммутаторов. Этой информации достаточно для корректирующей реконфигурации, а переход к синдрому неисправностей означает некоторую потерю информации: неисправности каналов относятся на счет ПЭ. Отсюда вытекают два пути дальнейшего развития обсуждаемой стратегии:

— развитие методов получения синдрома неисправностей по синдрому несогласия;

— развитие методов реконфигурации непосредственно по синдрому несогласия.

Первое направление широко известно как метод взаимопроверок в вычислительных системах. Подробности и библиографию можно найти в [5] и других работах того же автора. Пп. 1—4 обсуждаемой стратегии являются оригинальной параллельной версией алгоритма взаимопроверок.

Второе направление представляется перспективным, но выходит за рамки настоящей работы. Отметим только, что на этом пути не нужна никакая специальная аппаратура для самодиагностики. Процессорные элементы служат эталонами друг для друга. Критерием исправности является «согласие» всех ПЭ со своими логическими соседями в решетке требуемого типа. Задача состоит в поиске такой «согласованной» решетки в некоторой достаточно богатой избыточной решетке, часть элементов которой «не согласна» друг с другом по некоторым связям.

На рис. 4 представлена структура ячейки процессорной матрицы при программной реализации алгоритма реконфигурации. БСП и БУК заменяются блоком МК (i, j), подключенным к шине МК. Блок МК не взаимодействует с соседями. Коммутаторы снабжаются регистрами горизонтальной и вертикальной коммутаций (РГК и РВК) для запоминания микрокоманды коммутации. Логические координаты (i', j') поступают в регистры ПЭ (i, j).

Язык ЛОГИКА1 для описания клеточных автоматов. Язык ЛОГИКА1 предназначен для описания параллельного микропрограммного клеточного автомата и представляет собой язык алгебры логики, имеющий некоторые особенности. Индекс 1 является номером версии языка, предназначенной для описания комбинационных клеточных автоматов. По мере накопления опыта возникает полная версия языка ЛОГИКА.

Полное описание клеточного автомата нерационально, поскольку число элементов в матрице велико и не определено заранее. Однако в силу однородности структуры достаточно описать функции одной клетки и сигналы, поступающие на входы граничных элементов. В языке ЛОГИКА1 эта возможность обеспечивается тем, что каждая переменная имеет физические координаты (i, j). Над ними можно производить операции типа $(i \pm c_1, j \pm c_2)$, где c_1, c_2 —

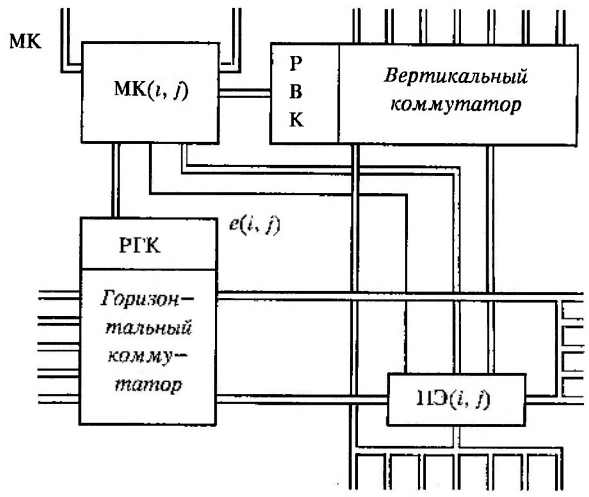

Рис. 4. Структура ячейки процессорной матрицы при программной реализации алгоритмов реконфигурации

константы. Это позволяет записывать булевы функции, зависящие от состояния самого элемента и его окрестности. Таким образом, в машинной памяти хранятся функции, описывающие только одну клетку, но характеризующие каждый элемент матрицы, что ведет к значительному сокращению затрат памяти. Кроме того, такое представление частично упорядочивает систему функций, облегчая тем самым процесс их вычисления. Заметим, что в отличие от машины клеточных автоматов [6] язык ЛОГИКА обеспечивает произвольные шаблон соседства, число состояний элементарного автомата, число переменных и при этом не требует для своей реализации специализированной ЭВМ.

Часто бывает необходимо задать особым образом функции граничных элементов. Минимальный номер строки и столбца в матрице — 0, а максимальный номер принято обозначать знаком «*». Так, элементы левой границы матрицы имеют физические координаты $(i, 0)$, правой — $(i, *)$, верхней — $(0, j)$, нижней — $(*, j)$, где $i = 0, ..., n - 1$, $j = 0, ..., m - 1$. Над этими индексами можно производить те же операции, что и над любыми другими.

Булева функция на языке ЛОГИКА задается произвольной скобочной формой. Допустимые в ней логические операции приведены в таблице в порядке возрастания приоритета.

Кроме физических индексов (i, j), каждый элемент имеет логические координаты (i', j'), первоначально совпадающие с физическими. В процессе реконфигурации логические координаты элемента изменяются. Вычисление логических индексов происходит аналогично вычислению коммутаций и производится клеточным автоматом, элементы которого выполняют как логические, так и арифметические операции. Для описания указанных клеточных

a	b	$a < b$	$a = b$	$a > b$	$a \$ b$	$a \wedge b$	$a \vee b$	$a \& b$	$\sim a$
0	0	0	1	0	1	0	0	0	1
0	1	1	0	0	1	1	1	0	1
1	0	0	0	1	0	1	1	0	0
1	1	0	1	0	1	0	1	1	0

автоматов в язык ЛОГИКА1 введены четыре основных арифметических действия над натуральными числами. Возникшая проблема совместимости арифметических и логических операций решена следующим образом. В языке ЛОГИКА1 булевы значения «ложь» и «истина» представлены числами 0 и 1. В арифметических операциях константы 0 и 1 рассматриваются как целые числа 0 и 1 соответственно. При выполнении логических операций целое число 0 выступает как логический 0, а натуральные числа интерпретируются как «не нуль», т. е. играют роль булевой константы 1. Приоритет логических операций выше, чем приоритет арифметических операций.

В качестве примера приведем запись на языке ЛОГИКА1 сигнала включения горизонтальной перемычки *dx* для алгоритма непосредственной перестройки [2]:

$$dx(i, j) : \sim ex'(i - 1, j) \& ex(i, j) \vee a(i, j) \& \sim ex(i, j) \&$$

$$\& [\sim ex(i - 1, j) \vee \sim ey(i - 1, j)];$$

Здесь «:» — знак присваивания значения.

Функции перенумерации для того же алгоритма имеют вид:

$$i'(i, j) : i'(i - 1, j) - dy(i, j) + 1;$$

$$j'(i, j) : j'(i, j - 1) - dx(i, j) + 1 - \sim ex(i, j) \& ey(i - 1, j) \& ex(i - 1, j) +$$

$$+ \sim ex(i, j) \& ey(i - 1, j - 1) \& \sim a(i - 1, j) + ex(i, j) \& ey(i, j) \& \sim ex(i - 1, j) -$$

$$- ex(i, j - 1) \& ey(i, j - 1) \& \sim ex(i - 1, j - 1);$$

Синтаксис языка ЛОГИКА1 задается далее в бекусовой форме. Поскольку некоторые символы этого метаязыка используются в языке ЛОГИКА, в бекусову форму внесены следующие изменения. Скобки ⟨,⟩ удалены из метаязыка. Термины, определяемые в метаязыке, являются словами без пробелов, а пробел разделяет эти конструкции и понимается как знак конкатенации. В этих обозначениях классическое определение идентификатора выглядит так:

идентификатор ::= буква | идентификатор буква | идентификатор цифра.

Пробел, используемый в языке ЛОГИКА1, в метаязыке изображается словом «пробел». Необязательные конструкции заключаются в фигурные скобки. Термин «ввод» соответствует клавише перевода курсора в начало следующей строки и используется для задания правил форматирования выражений языка ЛОГИКА1. Термин «файл» соответствует признаку конца файла.

Итак, метасимволы и термины метаязыка — это следующее множество (не считая запятых):

::=, |, {, }, ввод, пробел, , слово–без–пробела, файл.

Чтобы не загромождать описание языка сложными, но тривиальными конструкциями, используем содержательные определения. Последние отличаются тем, что в них встречаются термины, считающиеся известными без формального определения. Содержательное определение следует понимать как фразу обычного языка. Так, алфавит языка ЛОГИКА1 определяется содержательно:

алфавит ::= терминальные символы ЭВМ, кроме метасимволов.

Кроме того, в левой части бекусовой формы записываются синонимы, разделенные метасимволом |. Это позволяет дать понятиям равноправные со-

держательное и сокращенное наименования, например: НК | натуральная константа ::= ... .

Синтаксис языка ЛОГИКА1 задается следующими формами:

связка ::= & | ∨ | ∧ | $ | > | = | < | + | −
± ::= + | −
форматор ::= [|] | (|) | , | ; | пробел
разделитель ::= связка | форматор | : | ~ | *
цифра ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
просвет ::= пробел | просвет пробел
перенос ::= {просвет} ввод
НК | натуральная_константа ::= цифра | НК цифра
ЛК | логическая_константа ::= 0 | 1
бинар | бинарная_операция ::= {просвет} связка {просвет}
инверсия | унарная_операция ::= {просвет} ~{просвет}
и–символ ::= любой символ алфавита, кроме разделителя
имя ::= и–символ | имя и–символ | имя ∨
ik | i-координата ::= i {±НК} | * {±НК} | НК
jк | j-координата ::= j {±НК} | * {±НК} | НК
переменная ::= имя (ik, jк)
операнд ::= переменная | НК | ЛК
ф | формула ::= операнд | инверсия |ф| | [ф] бинар [ф]
выр | выражение ::= формула, в которой опущены внешние и некоторые
другие скобки так, что с учетом старшинства
операций порядок действий не нарушается
пч | правая_часть ::= выр | выр бинар перенос выр | выр перенос бинар выр
функция ::= переменная {просвет} : {просвет} правая_часть {просвет};
клетка ::= функция | клетка ввод функция
граница ::= НК, j | *{±НК}, j | i , НК | i, {±НК}
ГП | граничная_переменная ::= имя (граница)
ГУ | граничное_условие ::= ГП {просвет} : {просвет} ЛК {просвет};
край ::= ГУ | край ввод ГУ
КА | клеточный автомат ::= клетка край файл

Программное обеспечение реконфигурации матрицы. Процесс реконфигурации отказоустойчивой матрицы можно реализовать с помощью программ трансляции, интерпретации и визуализации.

Программа-транслятор преобразует описание автомата на языке ЛОГИКА1 в ДНФ логических функций. Последние представлены множеством троичных векторов из 64 компонент. Таким образом, система позволяет описывать и реализовывать автоматы, имеющие не более 64 переменных. Скобочная форма преобразуется в польскую инверсную запись на основе алгоритма Дейкстры. Каждый символ этой записи — переменная, константа или операция — представлен дуплетом двоичных векторов. Константы 0, 1 и булевы операции кодируются парой равных векторов всем 1. Пара векторов, соответствующая i-й переменной, строится следующим образом. Первый вектор содержит 0 в i-й компоненте и 1 в остальных компонентах. Второй вектор представляет собой инверсию первого. Будем рассматривать этот дуплет как некую конъюнкцию, причем комбинация «00» в i-й компоненте соответствует вхождению в конъюнкцию i-й переменной со знаком инверсии, комбинация «01» — вхождению данной переменной без инверсии, комбинация «10» — отсутствующим в конъюнкции переменным. Преобразовав польскую инверсную запись согласно правилам, выполним действия над конъюнкциями. Полученное в результате множество конъюнкций, представленных парами двоичных векторов, есть единичное множество исходной функции. Такое представление функции удобно для дальнейших действий.

Интерпретатор сканирует матрицу по координатам (i, j) и вычисляет значения всех функций, используя специальную троичную логику с множеством значений $\{0, 1, -\}$. Функции, вычисленные в процессе сканирования, имеют

значения {0, 1}, а невычисленные — {–}. Сканирование продолжается до тех пор, пока не будут вычислены все функции системы, т. е. все коммутации. Выбор режима интерпретации при вычислении функций обеспечивает простоту имитации и позволяет варьировать размеры матрицы без ретрансляции системы функций.

Описанная система имеет следующее применение: 1) вычисление функций реконфигурации матрицы в управляющей ЭВМ; 2) отладка новых алгоритмов реконфигурации структуры; 3) разработка и моделирование МПКА для иных приложений, например для самодиагностики матрицы.

В настоящее время описаны на языке ЛОГИКА и отлажены четыре алгоритма реконфигурации, приведенные в [2].

Заключение. Предложен метод программной реализации МПКА, управляющих реконфигурацией процессорной матрицы. Он позволяет удалить из системы неконтролируемые схемы и повысить выход годных.

СПИСОК ЛИТЕРАТУРЫ

1. **Сами М., Стефанелли Р.** Перестраиваемые архитектуры матричных процессорных СБИС // ТИИЭР. 1986. № 5.
2. **Воробьев В. А., Лаходынова Н. В.** Процессорная матрица с перестраиваемой структурой и перестраиваемым резервом // Автометрия. 1994. № 5.
3. **Ачасова С. В., Бандман О. Л.** Корректность параллельных вычислительных процессов. Новосибирск: Наука, 1990.
4. **Головкин Б. А.** Параллельные вычислительные системы. М.: Наука, 1980.
5. **Димитриев Ю. К.** Алгоритм самодиагностики вычислительных систем с программируемой структурой // Электронное моделирование. 1985. № 5.
6. **Тоффоли Т., Марголус Н.** Машины клеточных автоматов. М.: Мир, 1991.

Поступила в редакцию 10 ноября 1995 г.