ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ
INFORMATION-COMMUNICATION TECHNOLOGIES

УДК 004.272:004.021 **DOI: 10.24151/1561-5405-2019-24-1-51-63**

Отражение и транспонирование данных в матрице клеточно-автоматного вычислителя

И.В. Матюшкин, М.А. Заплетина

Институт проблем проектирования в микроэлектронике Российской академии наук, г. Москва, Россия

imatyushkin@niime.ru

Параллельные архитектуры вычислительных систем, в том числе с массивным параллелизмом, в настоящее время интересуют исследователей в области теоретической информатики. Основной задачей при этом становится аппаратное или алгоритмическое ускорение межпроцессорного обмена. Одним из подходов к построению алгоритмов может быть использование нетрадиционного формализма – нейронных сетей или клеточных автоматов (КА), реализующих модель ближнего взаимодействия элементарных вычислителей. В работе рассмотрены три операции с матричными данными: унарная (поэлементная), отражение, транспонирование. Операции реализованы параллельными алгоритмами в формализме КА в предположении, что данные введены в КА до начала расчета. Показано, что все представленные алгоритмы имеют линейную по размеру матрицы сложность. Движение и преобразование данных осуществлено с помощью введения в состояние ячейки битовых и/или тритовых, т.е. с тремя состояниями, флагов-компонент. Условия останова расчета – наступление stop-состояния флага выделенной ячейки КА или всех ячеек поля КА, т.е. их «заморозка». Освоение техники клеточно-автоматной алгоритмики на примере элементарных операций над матрицами может служить базой для решения более сложных задач (например, вычисление детерминанта матрицы), возникающих в рамках численного моделирования процессов, операций и устройств микроэлектроники.

Ключевые слова: клеточные автоматы; матрицы; линейная алгебра; параллельные вычисления; нетрадиционные архитектуры

Благодарности: работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 17-07-00570). Авторы благодарят студента МФТИ В.С. Кожевникова за участие в работе по улучшению алгоритмов транспонирования.

Для цитирования: Матюшкин И.В., Заплетина М.А. Отражение и транспонирование данных в матрице клеточно-автоматного вычислителя // Изв. вузов. Электроника. – 2019. – Т. 24. – № 1. – С. 51–63. DOI: 10.24151/1561-5405-2019-24-1-51-63

Data Reflection and Transposition in the Matrix of a Cellular Automata Computator

I.V. Matyushkin, M.A. Zapletina

Institute for Design Problems in Microelectronics of Russian Academy of Sciences, Moscow, Russia

imatyushkin@niime.ru

Abstract: The parallel architectures of computing systems, including the massively parallel ones, attract a particular interest of modern researchers in the theoretical informatics field. In this connection the hardware or algorithmic acceleration of the interprocessor exchange becomes the main objective. One of the approaches to creation of algorithms can be the use of nonconventional formalism-neural networks or cellular automata (CA), to realize the model of near interaction of elementary calculators. In the work three operations with matrix data have been considered: unary, reflection, transposing. The operations have been realized by parallel algorithms in the formalism of the cellular automata in an assumption that the data had been loaded into CA before the calculation. It has been shown that all presented algorithms have linear complexity of the matrix size. Movement and modification of the data have been executed by means of introducing the bit or/and trit flag components into a cell state description. The calculation stop-conditions are the occurrence of a stop-condition of a special CA cell or of all cells of the CA field, i.e. their «freezing». The development of the cellular automata algorithmization on an example of elementary operations over matrices can be used as a base for solving more difficult tasks (for example, the calculations of a determinant of a matrix).

Keywords: cellular automata; CA; matrix; linear algebra; parallel computing; nonconventional architectures

Acknowledgements: this study work has been funded by Russian Foundation for Basic Research (grant No. 17-07-00570). The authors also thank the student of MIPT V.S. Kozhevnikov for the participation in an improvement of the algorithms for transposing operation.

For citation: Matyushkin I.V., Zapletina M.A. Data reflection and transposition in the matrix of a cellular automata computator. *Proc. Univ. Electronics*, 2019, vol. 24, no. 1, pp. 51–63. DOI: 10.24151/1561-5405-2019-24-1-51-63

Введение. Нейроморфные вычисления, в частности в структуре клеточно-автоматного вычислителя, а также нейросетевые вычисления [1] демонстрируют неклассические принципы организации вычислений в ЭВМ, основанные на обучении сети путем подбора весов и порогов. Отличительные особенности таких вычислений – обращение к недетерминированности, что сближает их с квантовыми вычислениями, и преднамеренный отказ от контроля за состоянием памяти и путями прохождения информационного сигнала. Все это радикально меняет парадигму традиционного функционально-логического проектирования микросхем и, соответственно, создает множество проблем. Клеточно-автоматная архитек-

тура близка [2] к нейросетям, но принципы клеточно-автоматных вычислений подобны традиционной парадигме. По сравнению с традиционными методами распараллеливания [3, 4] алгоритмов и программ важен аспект отсутствия центрального процесса [2, 5] и невозможности прямой передачи данных между удаленными процессорными элементами, в роли которых выступают ячейки клеточных автоматов (КА). Отметим, что при разработке реального клеточно-автоматного вычислителя невозможно полностью добиться запрета ни центрального процесса, ни удаленного обмена между частями системы, но они выступают внешними по отношению к самому алгоритму вычислительного процесса, поэтому будем их считать запрещенными.

Рассмотрим массивный параллелизм, где число процессорных элементов превосходит число элементов данных, распределенных между последними. Цель настоящей работы – алгоритмизация простейших и примитивных операций в структуре такого клеточно-автоматного вычислителя. Другая интерпретация задачи, удобная для математиков-программистов, заключается в распараллеливании известных последовательных алгоритмов в условиях жестких ограничений, налагаемых моделью классического КА. На примерах примитивных задач можно показать «исчерпанность» старого подхода к вычислениям по отношению к нейроморфным вычислительным системам.

В [6] приведены примеры реализации средствами КА элементарных алгоритмов: сортировки символов (согласно алгоритму «чет-нечет» и идее блочного КА), сортировки строк (двумерная сортировка) и умножения натуральных чисел в произвольной системе счисления по модифицированной схеме Атрубина [7]. С точки зрения техники алгоритмизации задачи массовой унитарной операции и отражения матрицы относительно линии симметрии (если она диагональна, то говорят о транспонировании) элементарны, но значимы. Более сложные матричные задачи, например перемножения матриц, решались в 1960-е гг. для систолических процессорных массивов [7, 8] и в 1980-е гг. для процессорных матриц [9–11].

Принципиальным отличием предлагаемого подхода является ориентация на классическое математическое определение КА без привлечения каких-то модификаций, где постулируется изолированность КА, т.е. отсутствие «входов-выходов» [12] и, соответственно, обмена данных в процессе клеточно-автоматного вычисления. Единственное отклонение от классического определения КА – это конечность размера его поля. Относительно граничных условий КА преимущественно предполагается замыкание (схема тора) для локальной функции перехода (ЛФП), реже – два других варианта: линия «смерти» или естественное соседство. ЛФП для граничных ячеек выписываются особо в отличие от внутренних ячеек.

Важными инженерными вопросами при разработке клеточно-автоматного вычислителя являются подготовка исходных данных (считывание результата) и останов процесса. Они выходят за рамки настоящей работы. Условимся считать, что если глобальная конфигурации КА не изменяется на текущей итерации, то она является последней (останов по *idem*). Служебным словом *idem* будем также обозначать тот вариант ЛФП, при котором она не изменяет содержимого ячейки. Другим условием останова будем считать приведение одной, любой или выделенной, из ячеек КА в особое стоп-состояние (по аналогии со стоп-состоянием машины Тьюринга).

Итак, пусть заданы две квадратные матрицы размера $n \times n$: $A = \left\| a_{ij} \right\|_{1 \le i,\, j \le n}$. Перечислим примитивные задачи, требующие решения:

1) унарная поэлементная операция, прежде всего умножение на скаляр или сложение со скаляром: $a_{ij} \mapsto \lambda a_{ij}$, $a_{ij} \mapsto a_{ij} + \lambda$;

2) перестановка столбцов (или строк) в обратном направлении, т.е. зеркальное отражение относительно центральной оси: $a_{ij} \mapsto a_{i(n+1-j)}$;

3) транспонирование, т.е. отражение относительно главной диагонали: $a_{ij} \mapsto a_{ji}$.

Решение этих задач отнюдь не примитивно и требует изобретательности.

Следует отметить, что освоение способов выполнения простейших матричных операций на базе КА представляет собой одну из первых и необходимых ступеней использования формализма КА для этапов численного моделирования процессов, операций и устройств микроэлектроники. С этой точки зрения представленные простые клеточно-автоматные алгоритмы приобретают важный практический смысл.

Выполнение унарной поэлементной операции. В общем случае над всеми элементами матрицы реализуется одинаковая, возможно, параметризованная числом λ операция: $a_{ij} \mapsto f\left(a_{ij}, \lambda\right)$. В простейшем случае $f \doteq \lambda$ матрица заполняется одним числом. В качестве примера рассмотрим умножение на скаляр. Состояние ячейки задается триплетом $\langle b, p, s \rangle$, где $b \in \{0,1\}$ – булев флаг; p – регистр параметра (сюда помещается значение λ); s – регистр данных, содержащий a_{ij}. Слова «регистр» и «компонента (состояния)» синонимичны. На поле КА и шаблон окрестности не налагается спецификация кроме двумерности и единичности радиуса. Здесь и далее состояние самой ячейки индексировать не будем, а индексом α будем помечать любого ее соседа. Иногда будем использовать два индекса, например $s_{--} \equiv s_{i+1,j-1}$. Сокращенная и соответствующая ей развернутая индексация окрестности ячейки КА имеет следующий вид:

Сокращенная индексация			**Развернутая индексация**		
s_{-+}	s_{0+}	s_{++}	$s_{i-1,j-1}$	$s_{i-1,j}$	$s_{i-1,j+1}$
s_{-0}	s	s_{+0}	$s_{i,j-1}$	$s_{i,j}$	$s_{i,j+1}$
s_{--}	s_{0-}	s_{+-}	$s_{i+1,j-1}$	$s_{i+1,j}$	$s_{i+1,j+1}$

Начальные условия следующие:

$$\left(\forall i, j\right) : \left(s_{ij} = a_{ij}\right); \quad \left(\exists i', j' \in \{1, 2, ..., n\}\right) : \left(p_{i'j'} = \lambda\right) \wedge \left(b_{i'j'} = 1\right);$$
$$\left(\forall i \neq i', \forall j \neq j'\right) \cdot \left(p_{ij} = 0\right) \wedge \left(b_{i'j'} = 0\right).$$

Алгоритм сводится к размножению единиц в компоненте флага. ЛФП записывается в виде

$$\begin{aligned} &\left(b = 0\right) \wedge \left(\exists \alpha : b_\alpha = 1\right) \Rightarrow \left(b := 1\right) \wedge \left(p := p_\alpha\right) \wedge \left(s := p_\alpha s\right); \\ &\left(b = 1\right) \wedge \left(\forall \alpha : b_\alpha = 0\right) \Rightarrow \left(s := p_\alpha s\right); \\ &\left(b = 1\right) \wedge \left(\exists \alpha : b_\alpha = 1\right) \Rightarrow idem. \end{aligned} \qquad (1)$$

Знак «:=» означает присвоение, т.е. величина слева берется в момент $(t+1)$, а величины справа – в момент t.

Иногда вариант ЛФП, при котором состояние ячейки не изменяется, вообще не будем отдельно выписывать. В выражении (1) индекс α указывает на первый найденный индекс элемента с данным свойством. Очевидно, что временная сложность алгоритма не превосходит $2n$, где n – наибольший размер матрицы по одному из измерений. Возможно, при шаблоне Мура алгоритм завершит работу чуть раньше, чем при шаблоне окрестности Неймана.

Отражение относительно вертикали. Рассмотрим задачу отражения матрицы относительно вертикальной оси симметрии. Она легко обобщается для случая горизонтальной оси и сводится к отражению вектора длины n, например $(v_1, v_2, v_3, v_4, v_5) \to (v_5, v_4, v_3, v_2, v_1)$, $n = 5$. Очевидно, что в этом случае и КА одномерный. Простейшее решение основано на сортировке вектора индексов. Однако в клеточно-автоматной сортировке [3] останов возникает по *idem* для глобальной функции перехода. Здесь же приходится вводить дополнительно булеву компоненту состояния, чтобы реализовать останов по стоп-значению. Состояние ячейки – триплет $\langle a, b, s \rangle$, где $a \in \{0,1\}$ – флаг останова; $b \in \{0,1\}$ – флаг блочности; s – компонента данных. Границы естественные, т.е. у крайних ячеек сосед только один (для первой слева ячейки он индексируется через «+»). Начальное состояние имеет вид

$$(\forall i): (s_i = v_i); \quad a_1 = b_1 = 1; \quad (\forall i \neq 1): (a_i = 0) \wedge (b_i = \neg b_{i-1}).$$

Затем в слое флага останова единица будет смещаться вправо. При поднятии флага крайней правой ячейкой КА произойдет останов алгоритма. Следовательно, сложность алгоритма равна n. В слое флагов блочности будет наблюдаться чередование нулей и единиц со смещением вправо, активны будут ЛФП в блоке «01» (единица справа). Формула ЛФП задается набором последовательно выполняемых переходов (табл.1). Без условия перехода 6 и подготовительных условий перехода 1 и 2 алгоритм не имел бы внутренних, т.е. зависящих от глобальной конфигурации КА, средств останова.

Таблица 1

Локальная функция перехода для первого алгоритма отражения вектора

Table 1

Local rule for vector reflection algorithm

Номер перехода	Условие перехода	Формула перехода
1	$i \neq 1$: $a_- = 1$	$.a := 1.$
2	$a = 1$	$a := 0$
3	Всегда	$b := \neg b$
4	$i \neq n$: $(b = 0) \wedge (b_+ = 1)$	$s := s_+$
5	$i \neq 1$: $(b = 1) \wedge (b_- = 0)$	$s := s_-$
6	$i = n$: $a = 1$	*stop*

Рассмотрим второе клеточно-автоматное решение задачи, предполагающее замкнутость границ и останов по *idem*. Замкнутость поля означает, что индексы соседства первой и последней ячеек равны соответственно $(-,+)_1 \equiv (n,2)$, $(-,+)_n \equiv (n-1,1)$. Состояние ячейки КА – триплет $\langle s, m, b \rangle$, где s – компонента данных; $m \in \{-1,1\}$ – флаг подвижности (mobility), определяющий направление движения элемента вектора, содержащегося в клетке; $b \in \{0,1\}$ – флаг фиксации для реализации *idem* в ЛФП и останова алгоритма по *idem* в этой компоненте при $b = 1$.

Начальная конфигурация (тривиальный случай $n = 1$ не рассматривается) имеет вид

$$(\forall i): s_i = v_i; \; m_{n-1} = 1; \; (\forall i \neq n-1): m_i = -1; \; b_1 = b_n = 1;$$
$$(\forall i \notin \{1, n\}): b_i = 0.$$

Это позволяет после первой итерации работы КА получить первый и последний элементы вектора на их отраженных позициях.

Опишем ЛФП в виде таблицы условных переходов (табл.2). Для данных конфигураций логические условия не пересекаются, и поэтому срабатывания переходов независимы (не зависят от порядка). Условие 1 действует только на первой итерации и определяет перестановку двух крайних элементов вектора между собой, а также «замораживает» (фиксирует) ячейки, переводя (формула условия 3 табл.2) их в состояние *idem* ЛФП. Условие 2 реализует обмен данными между соседними ячейками КА. По условию 4 происходит формирование блокирующей комбинации (условие 3) для ячейки, в которой элемент вектора достиг своей окончательной позиции. Алгоритм завершит работу, когда все ячейки будут «заморожены». Пример его работы показан в табл.3. Сложность разработанного алгоритма, как показывает эксперимент, равна $2(n-2)$. Для $n=2$ требуются две итерации, для $n=3$ – три итерации.

Таблица 2

Локальная функция перехода для второго алгоритма отражения вектора с остановом по *idem*

Table 2

Local rule for vector reflection algorithm with *idem* stop

Номер перехода	Условие перехода	Формула перехода
1	$(b=1)\wedge(b_{-}=1)\wedge(m=-1)\wedge(m_{-}=-1)$	$s:=s_{-}, m:=1$
	$(b=1)\wedge(b_{+}=1)\wedge(m=-1)\wedge(m_{+}=-1)$	$s:=s_{+}, m:=1$
2	$(b=0)\wedge(b_{-}=0)\wedge(m=1)\wedge(m_{-}=-1)$	$s:=s_{-}, m:=1$
	$(b=0)\wedge(b_{+}=0)\wedge(m=1)\wedge(m_{+}=-1)$	$s:=s_{+}, m:=-1$
3	$(b=1)\wedge(m=1)$	*idem*
4	$(b_{-}=1)\wedge(m=1)\wedge(m_{-}=1)$	$b:=1$
5	$(b_{+}=1)\wedge(m=-1)\wedge(m_{+}=1)$	$m:=1$

Таблица 3

Динамика изменения поля КА с ЛФП, заданной табл.2, для вектора длиной 7

Table 3

Cellular automata (table 2 local rule) dynamics for vector with length of 7 elements

Номер итерации	Флаг *m*	Флаг *b*	Данные *s*
0	0 0 0 0 0 1 0	1 0 0 0 0 0 1	1 2 3 4 5 6 7
1	1 0 0 0 1 0 1	1 0 0 0 0 0 1	7 2 3 4 6 5 1
2	1 0 0 1 0 1 1	1 0 0 0 0 0 1	7 2 3 6 4 5 1
3	1 0 1 0 1 0 1	1 0 0 0 0 0 1	7 2 6 3 5 4 1
4	1 1 0 1 0 1 1	1 0 0 0 0 0 1	7 6 2 5 3 4 1
5	1 1 1 0 1 0 1	1 1 0 0 0 0 1	7 6 5 2 4 3 1
6	1 1 1 1 0 1 1	1 1 1 0 0 0 1	7 6 5 4 2 3 1
7	1 1 1 1 1 0 1	1 1 1 1 0 0 1	7 6 5 4 3 2 1
8	1 1 1 1 1 1 1	1 1 1 1 1 0 1	7 6 5 4 3 2 1
9	1 1 1 1 1 1 1	1 1 1 1 1 1 1	7 6 5 4 3 2 1

Примечание. Для *m*-флага нулю соответствует состояние «–1» (движение вправо), а единице – «+1» (движение влево).

Рассмотрим более быструю модификацию второго алгоритма. Состояние ячейки описывается триплетом $\langle s, m, b \rangle$, где s – компонента данных; $m \in \{-1, 1, 0\}$ – тритовый флаг подвижности; $b \in \{0, 1\}$ – битовый флаг фиксации. По аналогии с описанным алгоритмом комбинация флагов $(b := 1) \wedge (m := 0)$ является блокирующей для ЛФП, т.е. «замораживает» ячейку и обеспечивает останов по *idem* (табл.4).

Таблица 4

Локальная функция перехода для модификации второго алгоритма отражения вектора с остановом по *idem*

Table 4

Local rule for vector reflection algorithm with the *idem* stop

Номер перехода	Условие перехода	Формула перехода
1	$(m = -1) \wedge (m_+ = -1) \wedge (b = 1) \wedge (b_+ = 1)$	$s := s_+, m := 0$
	$(m = -1) \wedge (m_- = -1) \wedge (b = 1) \wedge (b_- = 1)$	$s := s_-, m := 0$
2	$(m = -1) \wedge (m_- = 0) \wedge (b \neq 1) \wedge (b_- \neq 1)$	$s := s_-, m := 0$
	$(m = 0) \wedge (m_+ = -1) \wedge (b \neq 1) \wedge (b_+ \neq 1)$	$s := s_+, m := -1$
3	$(m = 0) \wedge (m_- = 1) \wedge (b \neq 1) \wedge (b_- \neq 1)$	$s := s_-, m := 1$
	$(m = 1) \wedge (m_+ = 0) \wedge (b \neq 1) \wedge (b_+ \neq 1)$	$s := s_+, m := 0$
4	$(b = 1) \wedge (m = 0)$	*idem*
5	$(m = 0) \wedge (m_- = 0) \wedge (b \neq 1) \wedge (b_- = 1)$	$m := 1$
	$(m = 0) \wedge (m_+ = 0) \wedge (b \neq 1) \wedge (b_+ = 1)$	$m := -1$
6	$(m = -1) \wedge (m_- = 1) \wedge (b \neq 1) \wedge (b_- \neq 1)$	$s := s_-, m := m_-$
	$(m = 1) \wedge (m_+ = -1) \wedge (b \neq 1) \wedge (b_+ \neq 1)$	$s := s_+, m := m_+$
7	$(b \neq 1) \wedge ((m = 1) \wedge (m_+ = 0) \wedge (b_+ = 1) \vee$ $\vee (m = -1) \wedge (m_- = 0) \wedge (b_- = 1))$	$m := 0, b := 1$

Начальная конфигурация различается для четных и нечетных длин векторов:

$$n = 2p + 1: \quad (\forall i): s_i = v_i; \quad (\forall i): m_i = 0; \; b_{p+1} = 1;$$
$$(\forall i \neq p + 1): b_i = 0;$$
$$n = 2p: \quad (\forall i): s_i = v_i; \quad m_p = -1, \, m_{p+1} = -1;$$
$$(\forall i \neq p, \, p + 1): m_i = 0; b_p = 1, \, b_{p+1} = 1;$$
$$(\forall i \neq p, \, p + 1): b_i = 0.$$

Для вектора нечетной длины при инициализации достаточно указать фиксированный центральный элемент $(b = 1) \wedge (m = 0)$, относительно которого будет проводиться отражение, в то время как для вектора с четным числом элементов нужно на первой итерации «заморозить» и обменять данные центральных элементов (см. табл.4 условие 1). Поэтому для них специфицируются особые начальные условия по флагам $m = (-1, -1)$ и $b = (1, 1)$.

Принцип работы алгоритма состоит в том, что элементы вектора, находящиеся в соответствующих клетках КА, двигаются от центральных «замороженных» элементов через левый и правый края поля автомата к своим позициям в отраженном векторе. В первой фазе своего движения каждый элемент вектора доходит до центрального элемента своей половины поля КА и отталкивается от него, а во второй фазе он через край поля КА посте-

пенно перемещается к своей окончательной позиции. Опишем ЛФП КА в виде таблицы условных переходов (см. табл.4). Условия 2, 3, 6 реализуют обмен данными между соседними ячейками КА. Условия 5 задают изменение флага m ячеек, оказавшихся вблизи «замороженного» центра, имитируя их отталкивание от него. По условию 7 происходит формирование блокирующей комбинации условия 4 для ячейки, в которой элемент вектора достиг своей окончательной позиции. Алгоритм завершит работу, когда все ячейки будут «заморожены». Примеры его работы показаны в табл.5 и 6.

Сложность разработанного алгоритма, как показывает эксперимент, для четных длин вектора равна $1{,}5n-2$, для нечетных – $1{,}5(n-1)$.

Таблица 5

Динамика изменения поля КА с ЛФП, заданной табл.4, для вектора длиной 8

Table 5

Cellular automata (table 4 local rule) dynamics for vector with length of 8 elements

Номер итерации	Флаг *m*	Флаг *b*	Данные *s*
0	0 0 0 -1 -1 0 0 0	0 0 0 1 1 0 0 0	1 2 3 4 5 6 7 8
1	0 0 0 0 0 0 0 0	0 0 0 1 1 0 0 0	1 2 3 4 5 6 7 8
2	0 0 -1 0 0 1 0 0	0 0 0 1 1 0 0 0	1 2 3 4 5 6 7 8
3	0 -1 0 0 0 0 1 0	0 0 0 1 1 0 0 0	1 3 2 5 4 7 6 8
4	-1 0 -1 0 0 1 0 1	0 0 0 1 1 0 0 0	3 1 2 5 4 7 8 6
5	1 -1 0 0 0 0 1 -1	0 0 0 1 1 0 0 0	6 2 1 5 4 8 7 3
6	-1 1 -1 0 0 1 1 1	0 0 0 1 1 0 0 0	2 6 1 5 4 8 3 7
7	1 -1 1 0 0 -1 1 -1	0 0 0 1 1 0 0 0	7 1 6 5 4 3 8 2
8	-1 1 0 0 0 0 -1 1	0 0 1 1 1 1 0 0	1 7 6 5 4 3 2 8
9	1 0 0 0 0 0 0 -1	0 1 1 1 1 1 1 0	8 7 6 5 4 3 2 1
10	0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1	8 7 6 5 4 3 2 1

Таблица 6

Динамика изменения поля КА с ЛФП, заданной табл.4, для вектора длиной 7

Table 6

Cellular automata (table 4 local rule) dynamics for vector with length of 7 elements

Номер итерации	Флаг *m*	Флаг *b*	Данные *s*
0	0 0 0 0 0 0 0	0 0 0 1 0 0 0	1 2 3 4 5 6 7
1	0 0 -1 0 1 0 0	0 0 0 1 0 0 0	1 2 3 4 5 6 7
2	0 -1 0 0 0 1 0	0 0 0 1 0 0 0	1 3 2 4 6 5 7
3	-1 0 -1 0 1 0 1	0 0 0 1 0 0 0	3 1 2 4 6 7 5
4	1 -1 0 0 0 1 -1	0 0 0 1 0 0 0	5 2 1 4 7 6 3
5	-1 1 -1 0 1 -1 1	0 0 0 1 0 0 0	2 5 1 4 7 3 6
6	1 -1 1 0 -1 1 -1	0 0 0 1 0 0 0	6 1 5 4 3 7 2
7	-1 1 0 0 0 -1 1	0 0 1 1 1 0 0	1 6 5 4 3 2 7
8	1 0 0 0 0 0 -1	0 1 1 1 1 1 0	7 6 5 4 3 2 1
9	0 0 0 0 0 0 0	1 1 1 1 1 1 1	7 6 5 4 3 2 1

Транспонирование. Рассмотрим первый вариант алгоритма отражения матрицы относительно главной диагонали для поля размера $n \times n$ с шаблоном Мура (8 ячеек) и естественными границами (нет замыкания границ, для угловых ячеек всего 3 соседа). Состояние ячейки задается двумя компонентами $\langle a,s \rangle$, где s – компонента данных;

$a \in \{0,1,*\}$ – флаг перемещения, значения «0» и «1» которого соответствуют активным элементам, «*» – фиксированным.

Начальное состояние задается следующим образом и не содержит «*» во флаге *a*:

$$(\forall i, j): s_{i,j} = m_{i,j}, \quad b_{i,j=1} = b_{i=n,\mathrm{j}} = 1;$$
$$(\forall j \neq 1, i \neq n): a_{i,j} = \neg a_{i+1,j-1}.$$

Несмотря на то что флаг a может принимать три различных значения $\{0,1,*\}$, в формуле перехода 4 используется логическое отрицание $a := \neg a$ (табл.7). Это оправдано тем, что соответствующее условие исключает вариант $a = *$.

Таблица 7

Локальная функция перехода для первого варианта алгоритма транспонирования матрицы

Table 7

Local rule for the 1st matrix transposition algorithm

Номер перехода	Условие перехода	Формула перехода
1	$a = 2$	*idem*
2	$(i \neq n) \wedge (j \neq 1): a = 1$	$s := s_{--}$
	$(i \neq 1) \wedge (j \neq n): a = 0$	$s := s_{++}$
3	$(i = j) \wedge ((i = 1) \vee (i = n))$: *всегда*	$a := *$
	$(i \neq j) \vee ((i \neq 1) \wedge (i \neq n))$: $\left(\exists \alpha \in \{(-,0),(+,0),(0,+),(0,-)\}: a_\alpha = *\right)$	
4	$(i \neq j) \vee ((i \neq 1) \wedge (i \neq n))$: $\left(\forall \alpha \in \{(-,0),(+,0),(0,+),(0,-),(0,0)\}: a_\alpha \neq *\right)$	$a := \neg a$

Приведенный алгоритм (см. табл.7) производит перестановки элементов по диагонали (ячейка меняется с правым верхним либо с левым нижним соседом) аналогично первому алгоритму отражения одномерного массива (см. табл.1). Элементы матрицы, стоящие на линиях, параллельных побочной диагонали, образуют одномерные массивы, которые отражаются независимо. Существенен тот факт, что отражение происходит ровно за m шагов, где m – длина одномерного массива. Так как при транспонировании матрицы дольше всего отражается побочная диагональ, имеющая длину, равную порядку матрицы n, то и весь процесс занимает ровно n шагов. Останов происходит, когда флаги всех ячеек принимают значение $a = *$. На первом шаге это значение выставляется в граничных ячейках главной диагонали (в левой верхней и правой нижней), а затем копируется всеми ячейкам, но только по вертикали и горизонтали (не по диагонали). Остановочное значение флага доходит до главной диагонали за n шагов, что обеспечивает линейную сложность алгоритма.

Рассмотрим второй вариант алгоритма с иными характеристиками КА: окрестностью фон Неймана и замкнутыми границами поля. Состояние ячейки КА описывается четырьмя компонентами $\{s, a, b, f\}$, где s – компонента данных; $a \in \{0,1\}$ – флаг перемещения; $b \in \{0,1\}$ – флаг границы (выделяет часть клеток, которые не взаимодействуют с остальными, создает эффективную границу); $f \in \{0,1,*\}$ – флаг останова. Начальные условия задаются следующим образом:

$$(\forall i, j): s_{i,j} = m_{i,j}, \quad b_{i,j=1} = b_{i=1,j} = a_{i,j=1} = a_{i=1,j} = a_{i=n,j} = a_{i,j=n} = 1;$$
$$(\forall i \notin \{1,n\}, j \notin \{1,n\}): a_{i,j} = \neg a_{i+1,j+1};$$
$$(\forall i \neq 1, j \neq 1): b_{i,j} = 0;$$
$$f_{n,n} = 1;$$
$$(\forall (i,j) \neq (n,n)): f_{i,j} = 0.$$

Данный алгоритм также основан на алгоритме отражения (см. табл.1). Матрица мысленно разбивается на n одномерных массивов (рисунок). За исключением самого длинного их них, все такие массивы независимо отражаются указанным способом, что реализовано условием 1 (табл.8).

Таблица 8

Локальная функция перехода для второго варианта алгоритма транспонирования матрицы

Table 8

Local rule for the 2nd matrix transposition algorithm

Номер перехода	Условие перехода	Формула перехода
1	$(f \neq *) \wedge (a = 0) \wedge (a_{0+} = 1) \wedge (b = 0) \wedge (b_{0+} = 0)$	$s := s_{0+}$
	$(f \neq *) \wedge (f_{+0} \neq *) \wedge (a = 0) \wedge (a_{+0} = 1) \wedge (b = 0) \wedge (b_{+0} = 0)$	$s := s_{+0}$
	$(f \neq *) \wedge (f_{0-} \neq *) \wedge (a = 1) \wedge (a_{0-} = 0) \wedge (b = 0) \wedge (b_{0-} = 0)$	$s := s_{0-}$
	$(f \neq *) \wedge (a = 1) \wedge (a_{-0} = 0) \wedge (b = 0) \wedge (b_{-0} = 0)$	$s := s_{-0}$
2	$(f \neq *) \wedge (a = 1) \wedge (b = 1) \wedge (b_{0-} = 1)$	$s := s_{0-}$
	$(f \neq *) \wedge (a = 0) \wedge (b = 1) \wedge (b_{+0} = 1)$	$s := s_{+0}$
3	$f \neq *$	$a := \neg a$
4	$(f = 1) \vee ((f_{+0} = 1) \wedge (f_{0-} = 1))$	$f := *$
	$(f = 0) \wedge ((f_{+0} = *) \vee (f_{0-} = *))$	$f := 1$

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}	a_{57}
a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	a_{66}	a_{67}
a_{71}	a_{72}	a_{73}	a_{74}	a_{75}	a_{76}	a_{77}

Повекторное представление траспонируемой матрицы
The vector representation of the matrix to transpose

Самый длинный массив, состоящий из первых столбца и строки и образующий эффективную границу, сортируется следующим образом: на каждом шаге то столбец, то строка поочередно циклически сдвигаются соответственно вверх и влево, что обеспечивается условием 2 (см. табл.8). Ячейки этого массива имеют значение $b = 1$, в то время как у всех остальных $b = 0$. Это позволяет изолировать их друг от друга, что необходимо для предотвращения непредусмотренного передвижения элементов через край автомата. Условие 3 отвечает за чередование нулей и единиц во флаге a, который определяет направление движения элементов в соответствии с условиями 1 и 2.

Останов работы алгоритма происходит по значению «*» флага f, распространяющемуся из правого нижнего угла с задержкой в один шаг, что реализовано комбинацией условий 4 локальной функции перехода (см. табл.8). Полное транспонирование матрицы выполняется за $2n-1$ шагов (табл.9).

Таблица 9

Пример работы второго алгоритма транспонирования для матрицы с $n = 4$

Table 9

Cellular automata (table 8 local rule) dynamics for matrix transpose algorithm with $n = 4$

Номер итерации	*a*				*b*				*f*				*s*			
0	1	1	1	1	1	1	1	1	0	0	0	0	1	5	9	13
	1	1	0	1	1	0	0	0	0	0	0	0	2	6	10	14
	1	0	0	1	1	0	0	0	0	0	0	0	3	7	11	15
	1	1	1	1	1	0	0	0	0	0	0	1	4	8	12	16
1	0	0	0	0	1	1	1	1	0	0	0	0	2	5	9	13
	0	0	1	0	1	0	0	0	0	0	0	0	3	7	14	10
	0	1	1	0	1	0	0	0	0	0	0	0	4	6	15	11
	0	0	0	0	1	0	0	0	0	0	0	*	1	8	12	16
2	1	1	1	1	1	1	1	1	0	0	0	0	5	9	13	2
	1	1	0	1	1	0	0	0	0	0	0	0	3	14	7	10
	1	0	0	1	1	0	0	0	0	0	0	1	4	8	12	11
	1	1	1	0	1	0	0	0	0	0	1	*	1	6	15	16
3	0	0	0	0	1	1	1	1	0	0	0	0	3	9	13	2
	0	0	1	0	1	0	0	0	0	0	0	0	4	8	10	7
	0	1	1	0	1	0	0	0	0	0	*	*	1	14	11	12
	0	0	0	0	1	0	0	0	0	0	*	*	5	6	15	16
4	1	1	1	1	1	1	1	1	0	0	0	0	9	13	2	3
	1	1	0	1	1	0	0	0	0	0	1	1	4	10	8	7
	1	0	1	0	1	0	0	0	0	1	*	*	1	6	11	12
	1	1	0	0	1	0	0	0	0	1	*	*	5	14	15	16
5	0	0	0	0	1	1	1	1	0	0	0	0	4	13	2	3
	0	0	1	0	1	0	0	0	0	*	*	*	1	6	7	8
	0	1	1	0	1	0	0	0	0	*	*	*	5	10	11	12
	0	0	0	0	1	0	0	0	0	*	*	*	9	14	15	16
6	1	1	1	1	1	1	1	1	0	1	1	1	13	2	3	4
	1	0	1	0	1	0	0	0	1	*	*	*	1	6	7	8
	1	1	1	0	1	0	0	0	1	*	*	*	5	10	11	12
	1	0	0	0	1	0	0	0	1	*	*	*	9	14	15	16
7	0	0	0	0	1	1	1	1	*	*	*	*	1	2	3	4
	0	0	1	0	1	0	0	0	*	*	*	*	5	6	7	8
	0	1	1	0	1	0	0	0	*	*	*	*	9	10	11	12
	0	0	0	0	1	0	0	0	*	*	*	*	13	14	15	16

Заключение. В завершение представим сравнительные характеристики пяти предложенных алгоритмов отражения: первых трех – относительно вертикали, последних двух – относительно диагонали (табл.10). В условиях классического изолированного КА потребовалось 2-4 дополнительных бита и 4-8 элементарных формул локальных функций перехода для управления процессом отражения векторов и матриц, что, по-видимому, является компромиссным решением для достижения высокого быстродействия, так как все представленные алгоритмы характеризуются линейной сложностью.

Важность проведенной работы обоснована теоретическим интересом к клеточно-автоматному распараллеливанию вычислений определителя матрицы, обратной матрицы и спектра собственных значений (либо характеристического полинома). Логично приступать к этой проблеме, уже имея опыт решения более простых задач. В литературе известно клеточно-автоматное решение для операции перемножения матриц [10] с помощью открытого КА, которое нужно адаптировать для изолированного случая, в то время как рассмотренные операции отражения, а также унарная операция ранее не исследовались.

Таблица 10

Сравнение алгоритмов отражения

Table 10

Comparison of reflection and transposition CA algorithms

Количество флагов	Сложность	Условие останова	Замыкание границ
2 бит	n	По значению	Нет
2 бит	$2n - 4$	*idem*	Есть
1 бит + 1 трит	$1{,}5n - 2$ для $n = 2p$; $1{,}5n - 1{,}5$ для $n = 2p+1$;	*idem*	Есть
1 трит	n	*idem*	Нет
2 бит и 1 трит (шаблон Неймана)	$2n - 1$	*idem*	Есть

Накладные расходы на перепрограммирование матрицы клеточно-автоматного вычислителя для обеспечения логических условий ЛФП и на конфигурирование начального состояния могут оказаться существенными. В таком случае алгоритмы на базе КА с «входами-выходами» представляются более выигрышными по сравнению с изолированными КА. Алгоритмы для изолированных КА не зависят от деталей технической реализации и поэтому более универсальны и сохраняют свое значение, несмотря на вероятность оказаться вычислительно менее эффективными для решения конкретной задачи.

Литература

1. ***Van Schaik A., Delbruck T., Hasler J.*** Neuromorphic engineering systems and applications // Frontiers Media SA, 2015. – 182 p.

2. ***Матюшкин И.В.*** Коннекционистское расширение минимальной модели вычислений. Ч. 1 // Философские проблемы информационных технологий и киберпространства. – 2016. – Т. 11. – № 1. – С. 103–120.

3. ***Гергель В.П., Стронгин Р.Г.*** Основы параллельных вычислений для многопроцессорных вычислительных систем: учеб. пособие. – Н. Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. – 184 с.

4. ***Ефимов С.С.*** Обзор методов распараллеливания алгоритмов решения некоторых задач вычислительной дискретной математики // Математические структуры и моделирование. – 2007. – № 17. – С. 72–93.

5. ***Gavrilov S.V., Matyushkin I.V., Stempkovsky A.L.*** Computability via cellular automata // Scientific and Technical Information Processing. – 2017. – Vol. 44. – No. 5. – P. 1–15.

6. ***Матюшкин И.В., Жемерикин А.В., Заплетина М.А.*** Клеточно-автоматные алгоритмы сортировки строк и умножения целых чисел по схеме Атрубина // Изв. вузов. Электроника. – 2016. – Т.21. – №6. – С. 557–565.

7. ***Atrubin A.J.*** A one-dimensional real-time iterative multiplier // IEEE Trans. on Electronic Computers. – 1965. – Vol. EC-14. – No.3. – P. 394–399.

8. ***Варшавский В.И., Мараховский В.Б., Песчанский В.А., Розенблюм Л.Я.*** Однородные структуры. Анализ. Синтез. Поведение. – М.: Энергия, 1973. – С. 112–123.

9. ***Legendi T., Katona E.*** Megacell machine // Proc. of the International Conference on Vector and Parallel Processors in Computational Science III (25–28 August 1987). – 1987. – Vol.8. – Iss.2. – P. 195–199.

10. ***Katona E.*** Cellular algorithms for binary matrix operations // Lecture Notes in Computer Science: International Conf. on Parallel Processing, CONPAR 1981. – Berlin, Heidelberg: Springer, 1981. – Vol.111. – P. 205–209.

11. ***Stojanović N.M., Milovanović I.Ž., Stojčev M.K., Milovanović E.I.*** Matrix-vector multiplication on a fixed size unidirectional systolic array // Computers and Mathematics with Applications. – 2000. – Vol.40. – P. 1189–1203.

12. ***Болотов А.А., Кудрявцев В.Б., Подколзин А.С.*** Теория однородных структур. – М.: Наука, 1990. – 296 с.

Поступила в редакцию 19.08.2018. г.; после доработки 19.08.2018 г. принята к публикации 27.11.2018 г.

Матюшкин Игорь Валерьевич – кандидат физико-математических наук, ведущий исследователь отдела систем автоматизированного проектирования интегральных схем Института проблем проектирования в микроэлектронике Российской академии наук (Россия, 124681, г. Москва, г. Зеленоград, ул. Советская, д. 3), imatyushkin@niime.ru

Заплетина Мария Андреевна – инженер-исследователь отдела систем автоматизированного проектирования интегральных схем Института проблем проектирования в микроэлектронике Российской академии наук (Россия, 124681, г. Москва, г. Зеленоград, ул. Советская, д. 3), zapletina_mariya@mail.ru

References

1. Van Schaik A., Delbruck T., Hasler J. *Neuromorphic Engineering Systems and Applications.* Frontiers Media SA, 2015. 182 p.

2. Matyushkin I.V. Connectionism expansion of minimal model of computation. *Filosofskiye problemy informatsionnykh tekhnologiy i kiberprostranstva = Philosophical problems of IT and Cyberspace*, 2016, no. 1, vol. 11, pp. 103–120. (in Russian).

3. Gergel. V.P., Strongin R.G. *Basics of parallel computing for multiprocessor computing systems: a tutorial*. Nizhny Novgorod, Publishing house of the Nizhny Novgorod State University named after N.I. Lobachevsky, 2003. 184 p. (in Russian).

4. Efimov S.S. A review of parallel methods for solutions of some discrete computational mathematics problems. *Matematicheskiye struktury i modelirovaniye = Mathematical structures and modeling*, 2007, no. 17, pp. 72–93. (in Russian).

5. Gavrilov S.V., Matyushkin I.V., Stempkovsky A.L. Computability via Cellular Automata. *Scientific and Technical Information Processing*, 2017, vol. 44, no. 5, pp. 1–15.

6. Matyushkin I.V., Zhemerikin A.V., Zapletina M.A. Cellular automata algorithms for sorting of strings and integer numbers multiplication based on Atrubin's scheme. *Izvestiya vuzov. Elektronika = Proceedings of Universities. Electronics*, 2016, vol.21, no. 6, pp. 557–565. (in Russian).

7. Atrubin A.J. A One-Dimensional Real-Time Iterative Multiplier. *IEEE Trans. on Electronic Computers*, 1965, vol. EC-14, no.3, pp. 394–399.

8. Varshavskii V.I., Marahovskii V.B., Peschanskii V.A., Rosenblyum L.Ya. *Homogeneous structures. Analysis. Synthesis. Behavior*. Moscow, Energy Publ., 1973, pp. 112–123. (in Russian).

9. Legendi T., Katona E. Megacell machine. *Proceedings of the International Conference on Vector and Parallel Processors in Computational Science III*, 1987, vol. 8, iss. 2, pp. 195–199.

10. Katona E. Cellular Algorithms for Binary Matrix Operations. *International Conference on Parallel Processing, CONPAR 1981. Lecture Notes in Computer Science*, 1981, vol.111, pp. 205–209.

11. Stojanović N.M., Milovanović I.Ž., Stojčev M.K., Milovanović E.I. Matrix-vector Multiplication on a Fixed Size Unidirectional Systolic Array. *Computers and Mathematics with Applications*, 2000, vol.40, pp. 1189–1203.

12. Bolotov A.A., Kudryavcev V.B., Podkolzin A.S. *The theory of homogeneous structures*. Moscow, Science, 1990. 296 p. (in Russian).

Received 19.08.2018; Revised 19.08.2018; Accepted 27.11.2018.

Information about the authors:

Igor V. Matyushkin – Cand. Sci. (Phis.-Math.), Leading Researcher of the EDA Department, Institute for Design Problems in Microelectronics of Russian Academy of Sciences (Russia, 124365, Moscow, Zelenograd, Sovetskaya str., 3), imatyushkin@niime.ru

Mariya A. Zapletina – Research Engineer of the EDA Department, Institute for Design Problems in Microelectronics of Russian Academy of Sciences (Russia, 124365, Moscow, Zelenograd, Sovetskaya str., 3), zapletina_mariya@mail.ru