УДК 004.272;004.021 DOI 10.31114/2078-7707-2018-3-132-137

Клеточно-автоматный вычислительный параллелизм элементарных матричных операций

И.В. Матюшкин, М.А. Заплетина

Институт проблем проектирования в микроэлектронике РАН, г. Зеленоград

imatyushkin@niime.ru, zapletina_mariya@mail.ru

***Аннотация* — Алгоритмы классической клеточно-автоматной формализации для некоторых элементарных операций над векторами и матрицами: унарной поэлементной операции, задач отражения и транспонирования матрицы представлены в требованиях массового параллелизма, более жёстких по сравнению с крупнозернистым параллелизмом, без привязки к конкретной архитектуре вычислителя. Все алгоритмы имеют линейную по размеру матрицы сложность.**

***Ключевые слова* — клеточные автоматы, матрицы, линейная алгебра, параллельные вычисления, нетрадиционные архитектуры.**

I. Введение

Нейроморфные и нейросетевые вычисления [1] демонстрируют новые принципы организации вычислений, основанные на обучении сети путем подбора весов и порогов. С другой стороны, существующие реализации параллельных вычислений на многоядерных системах во многом укоренены в старой (фон Неймановской) парадигме. Очевидно, что старая парадигма перестаёт удовлетворять современным требованиям, а новая – ещё не вполне готова к этому. К нейросетям близка клеточно-автоматная (КА) архитектура [2], но принципы КА-вычислений ближе к традиционной парадигме. При разработке КА-модели вычислений (и, соответственно, функциональной структуры КА-вычислителя) неизбежно встают вопросы реализации массивного параллелизма для элементарных, низкоуровневых алгоритмов. Их решение может подсказать оптимальную функциональную, вплоть до аппаратной реализации, схему как для ячейки КА-вычислителя, так и для его архитектуры. Предполагается, что мы имеем дело с таким массивным параллелизмом, что число процессорных элементов превосходит число элементов данных.

По сравнению с традиционными методами распараллеливания [3,4] алгоритмов и программ существенен аспект отсутствия центрального процесса [2,5] и невозможности прямой передачи данных между удаленными процессорными элементами, в роли которых выступают ячейки КА. Таким образом, КА-распараллеливание протекает в более жёстких условиях. Ранее мы приводили [6] примеры реализации средствами КА элементарных алгоритмов: сортировки символов (согласно алгоритму «чёт-нечет» и идее блочного КА), сортировки строк (двумерная сортировка) и умножения натуральных чисел в произвольной системе счисления по модифицированной схеме Атрубина [7].

Важными инженерными вопросами при разработке КА-вычислителя являются подготовка исходных данных (считывание результата) и останов процесса. Они выходят за рамки нашего рассмотрения. Условимся считать, что если глобальное состояние КА не изменяется на текущей итерации, то она является последней; назовем такой вид останова – «*idem*». Другим условием останова будем считать приведение одной, любой или выделенной, из ячеек КА в особое стоп-состояние (по аналогии со стоп-состоянием машины Тьюринга). При формулировке КА-алгоритма мы должны указывать начальное состояние КА, и оно должно содержать все исходные данные, включая дополнительные (служащие для управления процессом). КА по классическому определению является замкнутой системой и не обменивается сигналами с внешней средой при вычислении. Асинхронные и блочные КА мы принципиально не используем, оставаясь в рамках классического КА. Относительно граничных условий КА мы преимущественно будем предполагать зацикливание (схема тора) для локальной функции перехода (ЛФП), реже – два других варианта: линия «смерти» или естественное соседство. ЛФП для граничных ячеек мы всегда будем выписывать особо (в отличие от внутренних ячеек), немного отходя от классики.

Многие элементарные (типовые) алгоритмы связаны с действиями над матрицами. КА-формализация именно матричных операций, на наш взгляд, является необходимым и достаточным условием универсальной вычислимости средствами КА. Пусть заданы две квадратные матрицы размера $n \times n$: $A = \left\| a_{ij} \right\|_{1 \le i, j \le n}$, $B = \left\| b_{ij} \right\|_{1 \le i, j \le n}$. Перечислим элементарные 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: $a_{ij} \mapsto a_{ji}$.

4) Умножение матрицы на вектор-столбец (строку), т.е. $b_i \mapsto \sum_{j=1}^{n} a_{ji} b_j$.

5) Умножение матрицы на матрицу, т.е. $A \cdot B = \sum_{k=1}^{n} a_{ik} b_{kj}$.

6) Вычисление обратной матрицы $A^{-1} = E \mid A$.

7) Вычисление определителя $\det A$.

Данная статья посвящена поиску КА-алгоритмов для первых трёх задач. Отметим, что для трёх последних задач известны параллельные алгоритмы [6,8] (метод Фокса и ленточный для перемножения матриц), однако отсутствуют КА-формализации.

II. Выполнение унарной поэлементной операции

В общем случае мы реализуем над всеми элементами матрицы одинаковую, возможно, параметризованную числом λ, операцию: $a_{ij} \mapsto f\left(a_{ij}, \lambda\right)$. Простейшим случаем $f \doteq \lambda$ является заполнение матрицы 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)» для нас синонимичны. На поле КА и шаблон окрестности не налагается спецификации помимо двумерности и единичности радиуса. Здесь и далее состояние самой ячейки индексировать не будем, а индексом α будем помечать любого её соседа. Иногда будем использовать два индекса, например, $s_{(i-1)(j-1)} \equiv s_{--}$, $s_{(i+1)j} \equiv s_{+0}$, $s_{00} \equiv s$. Начальные условия:

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

Идея алгоритма сводится к размножению единиц в компоненте флага. Здесь и далее знак «:=», как обычно в программировании, означает присвоение, т.е. величина слева берется в момент (t+1), а величины справа – в момент t. ЛФП записывается просто:

$$\left(b = 0\right) \wedge \left(\exists \alpha : b_\alpha = 1\right) \Rightarrow \left(b := 1\right) \wedge \left(p := p_\alpha\right) \wedge \left(s := s \cdot p_\alpha\right);$$
$$\left(b = 1\right) \Rightarrow idem. \qquad (2)$$

С этого момента служебным словом «*idem*» будем отмечать не только глобальный останов, но и тот вариант ЛФП, при котором состояние ячейки не изменяется. Иногда такие варианты вообще не будем отдельно выписывать. В выражении (2) индекс α указывает на первый найденный индекс элемента, обладающего 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.

III. Отражение

Рассмотрим задачу отражения матрицы относительно вертикальной оси симметрии. Она легко обобщается для случая горизонтальной оси и сводится к отражению вектора длины n, т.е., например, $\left(v_1, v_2, v_3, v_4, v_5\right) \to \left(v_5, v_4, v_3, v_2, v_1\right)$, $n = 5$. Очевидно, в этом случае и КА одномерный. Простейшее решение основано на сортировке вектора индексов. Однако в КА-сортировке [3] останов возникает естественно – по условию *idem* для глобальной функции перехода. Здесь же нам 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:

$$\left(\forall i\right)\left(s_i = v_i\right), \quad a_1 = b_1 = 1,$$
$$\left(\forall i \neq 1\right)\left(a_i = 0\right) \wedge \left(b_i = \neg b_{i-1}\right). \qquad (3)$$

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 табл. 1 последовательно выполняемых правил.

Таблица 1

ЛФП для алгоритма отражения вектора

№	Условие перехода	Формула перехода
1	$i \neq 1 \quad \left(a_- = 1\right)$	$a := 1$
2	$a = 1$	$a := 0$
3	Всегда.	$b := \neg b$
4	$i \neq n \quad \left(b = 0\right) \wedge \left(b_+ = 1\right)$	$s := s_+$
5	$i \neq 1 \quad \left(b = 1\right) \wedge \left(b_- = 0\right)$	$s := s_-$
6	$i = n \quad \left(a = 1\right)$	*idem*

Без условия 6 и подготовительных условий 1 и 2 алгоритм не имел бы внутренних, т.е. зависящих от глобальной конфигурации КА, средств останова. Даже при срабатывании условия 6 переход к останову по *idem* фактически означал бы необходимость решения задачи залпового огня (fire squad), где «генерал» находился в правом конце «шеренги».

Поэтому рассмотрим другое КА-решение, 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) элемента вектора, содержащегося в клетке, определяющий направление его движения; $b \in \{0,1\}$ – флаг фиксации для реализации idem в ЛФП и останова алгоритма по idem в этой компоненте. Начальная конфигурация (тривиальный случай n=1 не рассматриваем):

$$\begin{aligned} &(\forall i)(s_i = v_i), \quad m_{n-1} = 1, (\forall i \neq n-1)(m_i = -1), \\ &b_1 = b_n = 1, \ (\forall i \notin \{1,n\})(m_i = -1), \end{aligned} \tag{1}$$

что позволяет после первой же итерации работы КА получить первый и последний элементы вектора на их отражённых позициях.

Таблица 2

Локальная функция перехода для алгоритма отражения вектора с остановом по idem

№	Условие перехода	Формула перехода
1	$(b=1)\wedge(b_-=1)\wedge$ $\wedge(m=-1)\wedge(m_-=-1)$	$s := s_-,\ m := 1$
	$(b=1)\wedge(b_+=1)\wedge$ $\wedge(m=-1)\wedge(m_+=-1)$	$s := s_+,\ m := 1$
2	$(b=0)\wedge(b_-=0)\wedge$ $\wedge(m=-1)\wedge(m_-=1)$	$s := s_-,\ m := 1$
	$(b=0)\wedge(b_+=0)\wedge$ $\wedge(m=1)\wedge(m_+=-1)$	$s := s_+,\ m := -1$
3	$(b=1)\wedge(m=1)$	*idem*
4	$(b_-=1)\wedge(m=1)\wedge(m_-=1)$	$b := 1$
5	$(b_+=1)\wedge(m=-1)\wedge(m_+=1)$	$m := 1$

Опишем ЛФП в виде таблицы условных переходов (см. табл. 2). Для данных конфигураций логические условия не пересекаются, и поэтому срабатывания переходов независимы (не зависят от порядка). Правило №1 действует только на первой итерации и определяет перестановку двух крайних элементов вектора между собой, а также «замораживает» (фиксирует) ячейки, переводя их в состояние *idem* ЛФП (№3, табл. 1). Правило №2 реализует обмен данных между соседними ячейками КА. По правилу №4 происходит формирование блокирующей комбинации (№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 2 итерации, для $n=3$ – 3 итерации.

Таблица 3

Динамика изменения КА с ЛФП, заданной в табл. 2, для вектора длины 7

№	Флаг *m*	Флаг *b*	Данные *s*
0	0000010	1000001	1234567
1	1000101	1000001	7234651
2	1001011	1000001	7236451
3	1010101	1000001	7263541
4	1101011	1000001	7625341
5	1110101	1100001	7652431
6	1111011	1110001	7654231
7	1111101	1111001	7654321
8	1111111	1111101	7654321
9	1111111	1111111	7654321

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

Рассмотрим более быструю модификацию второго алгоритма. Состояние ячейки описывается триплетом $\langle s,m,b\rangle$, где s – компонента данных, $m \in \{-1,1,0\}$ – флаг подвижности, а $b \in \{0,1\}$ – флаг фиксации: по аналогии с алгоритмом, показанным выше, комбинация флагов $(a := 1)\wedge(m := 0)$ является блокирующей для ЛФП и обеспечивает останов по *idem*. Начальная конфигурация различается для чётных и нечётных длин векторов:

$$\begin{aligned} &n = 2p+1: \ (\forall i)(s_i = v_i), \quad (\forall i)(m_i = 0), \\ &\quad a_{p+1} = 1, \quad (\forall i \neq p+1)(a_i = 0); \\ &n = 2p: \ (\forall i)(s_i = v_i), \quad m_p = -1, m_{p+1} = -1, \\ &(\forall i \neq p, p+1)(m_i = 0), \\ &a_p = 1, a_{p+1} = 1, (\forall i \neq p, p+1)(a_i = 0). \end{aligned} \tag{2}$$

Для вектора нечётной длины при инициализации достаточно указать фиксированный центральный элемент $(a=1)\wedge(m=0)$, относительно которого будет производиться отражение, в то время как для вектора с чётным числом элементов нужно на первой итерации «заморозить» и обменять данные центральных элементов (см. условия 1 табл. 4). 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 отражённом 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 постепенно перемещается к своей окончательной позиции.

Таблица 4

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

№	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	$(m=0)\wedge(m_{-}=0)\wedge(b_{-}=1)$	$m:=1$
	$(m=0)\wedge(m_{+}=0)\wedge(b_{+}=1)$	$m:=-1$
6	$(m=-1)\wedge(m_{-}=1)\wedge(b\neq 1)\wedge(b_{-}\neq 1)$	$s:=s_{-}, m:=1$
	$(m=1)\wedge(m_{+}=-1)\wedge(b\neq 1)\wedge(b_{+}\neq 1)$	$s:=s_{+}, m:=-1$
7	$(m=1)\wedge(m_{+}=0)\wedge(b_{+}=1)\vee$	$m:=0, b:=1$

Сложность разработанного алгоритма, как показывает эксперимент, для чётных длин вектора равна $1.5n-2$, для нечётных - $1.5(n-1)$.

Таблица 5

Динамика изменения КА, заданного ЛФП Таблицы 4, для вектора длины 8

№	Флаг *m*	Флаг *b*	Данные *s*
1	2	3	4
0	000-1-1000	00011000	12345678
1	00000000	00011000	12345678
2	00-100+100	00011000	12345678
3	0-10000+10	00011000	13254768
4	-10-100+10+1	00011000	31254786
5	+1-10000+1-1	00011000	62154873
6	-1+1-100+11+1	00011000	26154837
7	+1-1+100-1+1-1	00011000	71654382
8	-1+10000-1+1	00111100	17654328
9	+1000000-1	01111110	87654321
10	00000000	11111111	87654321

Примечание: Для *m*-флага значение 1 обозначается «+1».

Опишем ЛФП КА в виде таблицы условных переходов (см. табл. 4). Правила №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 правилу №7 происходит формирование блокирующей комбинации (№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 табл. 6.

Таблица 6

Динамика изменения поля КА, заданного ЛФП табл. 4, для вектора длины 7

№	Флаг *m*	Флаг *a*	Данные *s*
0	0000000	0001000	1234567
1	00-10+100	0001000	1234567
2	0-1000+10	0001000	1324657
3	-10-10+10+1	0001000	3124675
4	+1-1000-1-1	0001000	5214763
5	-1+1-10+1-1+1	0001000	2514736
6	+1-1+10-1+1-1	0001000	6154372
7	-1+1000-1+1	0011100	1654327
8	+100000-1	0111110	7654321
9	0000000	1111111	7654321

Примечание: расшифровка значения «+1» флага – аналогично табл. 5.

IV. Транспонирование

Поскольку транспонирование является отражением квадратной матрицы относительно её главной диагонали, естественно использовать идею алгоритма отражения (табл. 1). Рассмотрим вариант алгоритма 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 описывается четвёркой $\langle a,b,f,s\rangle$, где $a\in\{0,1\}$ – флаг динамики, $b\in\{0,1\}$ – флаг блочности, $f\in\{0,1,*\}$ – флаг (трит) прогресса, s – компонента данных. Начальное состояние задается (6):

$$\begin{aligned}&(\forall i,j)\left(s_{i,j}=m_{i,j}\right);\quad a_{i,j=1}=a_{i=n,j}=b_{i,j=1}=b_{i=n,\mathrm{j}}=1;\\&f_{i=1,\mathrm{j}}=f_{i,j=n}=1,\\&(\forall j\neq 1, i\neq n)\left(a_{i,j}=0\right)\wedge\left(b_{i,j}=\neg b_{i-1,j-1}\right)\wedge\left(f_{i,j}=0\right).\end{aligned}\qquad(3)$$

Таблица 7

ЛФП для алгоритма транспонирования матрицы

№	Условие перехода	Формула перехода
1	$((j\neq 1)\vee(i\neq n))\wedge(a_{--}=1)$	$a:=1$
2	$(f\neq *)\wedge(a=1)$	$a:=0$
3	$(f\neq *)$	$b:=\neg b$
4	$(b=0)\wedge(b_{++}=1)\wedge$	$s:=s_{++}$
5	$(b=1)\wedge(b_{--}=0)\wedge$	$s:=s_{--}$
6	$(f_{++}\neq 0)\wedge(a_{++}=1)\wedge(b_{++}=1)$	$a:=1,\ b:=1,\ f:=1$
7	$(a=1)\wedge(b=1)\wedge(f=1)$	$f:=*$

В процессе работы алгоритма (табл. 7) в слое флага динамики единицы из первого столбца и последней строки матрицы будут смещаться вверх и вправо до тех пор, пока не достигнут своих позиций в верхней строке и последнем столбце. В слое флагов блочности будет наблюдаться чередование нулей и единиц со смещением также вверх и вправо; активны будут ЛФП

в блоке $\langle b, b_{++} \rangle = \langle 0,1 \rangle$ (единица сверху справа). В слое флагов прогресса единицы будут по мере упорядочения элементов данных исходной матрицы заменяться на состояние заморозки «*», а нули – на единицы.

Таблица 8

Динамика изменения поля КА, заданного ЛФП табл. 7, для матрицы размера 4×4

№	Флаг *a*	Флаг *b*	Флаг *f*	Данные *s*
0	1000 1000 1000 1111	1010 1011 1000 1111	1111 0001 0001 0001	1 2 3 4 5 6 7 8 910 11 12 1314 15 16
1	1100 0100 0111 0001	1101 0100 0111 0001	*111 0001 0001 000*	1 2 6 4 5 3 10 11 9 7 8 12 1314 15 16
2	1110 1011 0001 0011	1110 1011 1001 1111	**11 1001 000* 001*	1 5 6 10 2 9 4 11 313 14 15 7 8 12 16
3	1111 1101 0001 0011	1111 1101 0111 0011	***1 *10* 001* 00**	1 5 9 10 2 6 13 14 3 4 11 15 7 8 12 16
4	1111 1111 1011 0111	1111 1111 1011 1111	**** **1* 10** 01**	1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
5	1111 1111 1111 0111	1111 1111 1111 0111	**** **** *1** 0***	1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
6	1111 1111 1111 1111	1111 1111 1111 1111	**** **** **** 1***	1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
7	1111 1111 1111 1111	1111 1111 1111 1111	**** **** **** ****	1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16

Глобальный останов работы КА происходит, когда флаги прогресса всех ячеек равны «*» (при этом значения остальных флагов могут быть любыми). Пример работы алгоритма для матрицы размера 4×4 представлен в табл. 8, а его сложность, как показал эксперимент, равна $2n-1$, для матрицы с $n = 2$ необходимо 2 итерации.

V. Заключение

В статье приведены отсутствующие в литературе строгие формулировки алгоритмов поэлементной унарной операции, отражения и транспонирования для матриц, элементы которых распределены по полю двумерного классического клеточного автомата. Все алгоритмы имеют первый порядок сложности по линейному размеру матрицы. В дальнейшем нами планируется алгоритмизировать КА-средствами более сложные, значимые для вычислительной математики проблемы: перемножение двух матрицы, расчёт определителя и собственных чисел.

Поддержка

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований: грант № 17-07-00570.

Литература

[1] van Schaik A. Neuromorphic Engineering Systems and Applications / van Schaik A., Delbruck T., Hasler J. – Frontiers Media SA, 2015. – 182 pages.

[2] Матюшкин И.В. Коннекционистское расширение минимальной модели вычислений (Часть 1) / Философские проблемы информационных технологий и киберпространства, 2016. № 1, Т. 11. – С.103–120.

[3] Гергель В.П. Основы параллельных вычислений для многопроцессорных вычислительных систем: учебное пособие. / Гергель В.П., Стронгин Р.Г. – Нижний 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. 2017. Vol. 44. № 5. Pp. 1–15.

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

[7] Atrubin A.J. A One-Dimensional Real-Time Iterative Multiplier. - IEEE Trans. on Electronic Computers, Vol. EC-14, No.3, June 1965, pp. 394-399.

[8] Choi J. A new parallel matrix multiplication algorithm on distributed-memory concurrent computers // High Performance Computing on the Information Superhighway, 1997. HPC Asia '97, proceedings. – 1997. – PP.224-229.

Cellular Automata Computational Parallelism of Elementary Matrix Operations

I.V. Matyushkin, M.A. Zapletina

Institute for Design Problems in Microelectronics of RAS, Zelenograd

zapletina_mariya@mail.ru

***Abstract* — The algorithms of strict classical cellular automata formalization with initial states declaration, local rule and neighborhood definitions for some elementary operations on vectors and matrices under the requirements of massive parallelism are presented. The dynamics of cellular automata field state is presented clearly. Among the tasks are the unary element operation, the matrix reflection and transposition. The original solutions meet the more stringent parallelization conditions for a computer with the cellular automata architecture in comparison to traditional methods. There is no reference to particular hardware model that makes the represented algorithms more universal. The methods for matrix reflecting and transposition introduce special global break criteria based on compositions of state flags of local cells and their first-order neighbors. The break on cellular automata state invariance was considered. All the algorithms have complexity** $O(n)$ **according to linear dimension of the matrix.**

***Keywords* — cellular automata, matrix operations, linear algebra, parallel computing, non-traditional architectures.**

REFERENCES

[1] van Schaik A. Neuromorphic Engineering Systems and Applications / van Schaik A., Delbruck T., Hasler J. – Frontiers Media SA, 2015. – 182 pages.

[2] Matjushkin I.V. Konnekcionistskoe rasshirenie minimal'noj modeli vychislenij (Chast' 1) / Filosofskie problemy informacionnyh tehnologij i kiberprostranstva, 2016. № 1, Vol.11. – P.103–120.

[3] Gergel' V.P. Osnovy parallel'nyh vychislenij dlja mnogoprocessornyh vychislitel'nyh sistem: uchebnoe posobie. / Gergel' V.P., Strongin R.G. – Nizhnij Novgorod: Izd-vo NNGU im. N.I. Lobachevskogo, 2003. – 184 pages.

[4] Efimov S.S. Obzor metodov rasparallelivanija algoritmov reshenija nekotoryh zadach vychislitel'noj diskretnoj matematiki // Matematicheskie struktury i modelirovanie, 2007. – № 17. – PP. 72–93.

[5] Gavrilov S.V., Matyushkin I.V., Stempkovsky A.L. Computability via Cellular Automata. Scientific and Technical Information Processing. - 2017. - Vol. 44. № 5. - PP. 1–15.

[6] Matjushkin I.V., Zhemerikin A.V., Zapletina M.A. Cellular Automata Algorithms for String Sorting and Integer Multiplication According to the Atrubin Scheme // Russian Microelectronics. – Vol.46, Is.7. – 2017. – PP.500-505.

[7] Atrubin A.J. A One-Dimensional Real-Time Iterative Multiplier. - IEEE Trans. on Electronic Computers, Vol. EC-14, No.3, June 1965, pp. 394-399.

[8] Choi J. A new parallel matrix multiplication algorithm on distributed-memory concurrent computers // High Performance Computing on the Information Superhighway, 1997. HPC Asia '97, proceedings. – 1997. – PP.224-229.