Оригинальная статья / Original article

УДК 004.031: 681.3

https://doi.org/10.21869/2223-1560-2025-29-3-137-156

Математические модели и схемы конвейерной обработки унитарных кодов в однородных вычислительных системах

Е.А. Титенко [1] ✉, А. С. Сизов [1], М. А. Титенко [1]

[1] Юго-Западный государственный университет
ул. 50 лет Октября, д. 94, г. Курск 305040, Российская Федерация

✉ e-mail: johntit@mail.ru

Резюме

Целью исследования *является создание моделей и конвейерных схем для высоко-производительной обработки унитарных кодов в однородных вычислительных системах.*

Методы *исследования основаны на теории проектирования однородных вычислительных систем, методах синтеза итеративных сетей и систем искусственного интеллекта. Унитарные коды являются сигнально-информационной базой для анализа и планирования параллельных процессов в однородных вычислительных системах. Известные одномерные и двумерные итеративные сети являются основой для создания однородных конвейерных схем и рекуррентных вычислений в них. Тем не менее итеративные сети, состоящие из однородных вычислительных ячеек с регулярными связями, по умолчанию реализуют единственный вычислительный процесс и, как правило, одну поисково-вычислительную функцию Для повышения удельной производительности работы конвейерных схем развиты принципы мульти функциональности и мульти конвейеризации, позволяющие реализовывать на каждом такте работы конвейера работу нескольких ячеек, реализующих более одной операции.*

Результаты. *Созданы практически значимые конвейерные схемы с организацией нескольких локальных вычислительных процессов, имеющих собственные стартовые точки, что необходимо для эффективной работы однородных вычислительных систем – реконфигурируемые вычислительные структуры, машины баз данных и знаний, ассоциативные процессоры и др. Сравнительные оценки конвейерных схем проведены для общезначимых операций обработки унитарных кодов: арбитраж, формирование правой, левой серии логических «1». Показано, что применение принципов мультифункциональности и мультиконвейеризации обеспечивает пропорциональное уменьшение времени, приходящегося на одну операцию.*

Заключение. *Синтез параллельно-конвейерных схем обработки унитарных кодов на базе итеративных сетей основан на объединении и развитии принципов тактирования ячеек, мультифункциональности ячеек, мультиконвейеризации, что позволяет вести эффективную обработку потоков унитарных кодов, имеющих дуальную трактовку элементов (цифра/символ).*

Ключевые слова: *итеративная сеть; вычислительная ячейка; мультифункциональность сети; мультиконвейеризация; рекуррентная формула.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Для цитирования: Титенко Е.А., Сизов А. С., Титенко М. А. Математические модели и схемы конвейерной обработки унитарных кодов в однородных вычислительных системах // Известия Юго-Западного государственного университета. 2025; 29(3): 137-156. https://doi.org/ 10.21869/2223-1560-2025-29-3-137-156.

Поступила в редакцию 03.06.2025 *Подписана в печать 22.08.2025* *Опубликована 30.09.2025*

Mathematical models and schemes of conveyor processing of unitary codes in homogeneous computing systems

Evgeny A. Titenko [1] ✉, Alexandr S. Sizov [1], Mikhail A. Titenko [1]

[1] Southwest State University
50 Let Oktyabrya str. 94, Kursk 305040, Russian Federation

✉ e-mail: johntit@mail.ru

Abstract

***Purpose**.of the work is to create models and pipeline schemes for high-performance processing of unitary codes in homogeneous computing systems.*
The research methods are based on the theory of designing homogeneous computing systems, methods of synthesis of iterative networks and artificial intelligence systems. Unitary codes are a signal-information base for analyzing and planning parallel processes in homogeneous computing systems. Known one-dimensional and two-dimensional iterative networks are the basis for creating homogeneous pipeline schemes and recurrent computations in them.
***Methods.** Nevertheless, iterative networks consisting of homogeneous computing cells with regular connections, by default implement a single computing process and, as a rule, one search and computing function. To increase the specific performance of pipeline schemes, the principles of multi-functionality and multi-pipelining have been developed, allowing the implementation of several cells implementing more than one operation at each cycle of the pipeline.*
***Results.** Practically significant pipeline schemes with the organization of several local computing processes with their own starting points have been created, which is necessary for the efficient operation of homogeneous computing systems - reconfigurable computing structures, database and knowledge machines, associative processors, etc.*
Comparative assessments of pipeline schemes have been carried out for generally significant operations of processing unitary codes: arbitration, formation of the right, left series of logical "1". It is shown that the application of the principles of multi-functionality and multi-pipelining provides a proportional decrease in the time per operation.
***Conclusion.** The synthesis of parallel-pipeline schemes for processing unitary codes based on iterative networks is based on the unification and development of the principles of cell clocking, multi-functionality of cells, multi-pipelining, which allows for efficient processing of unitary code flows with dual interpretation of elements (digit/symbol).*

***Keywords:** iterative network; computational cell; multi-functionality of the network; multi-pipelining; recurrent formula.*

***Conflict of interest:** The Authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

For citation: Titenko E. A., Sizov A. S., Titenko M. A. Mathematical models and schemes of conveyor processing of unitary codes in homogeneous computing systems. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2025; 29(3): 137-156 (In Russ.). https://doi.org/ 10.21869/2223-1560-2025-29-3-137-156.

Received 03.06.2025 *Accepted 22.08.2025* *Published 30.09.2025*

***

Введение

Однородные вычислительные системы (ОВС) [1, 2] представляют класс высокопроизводительных ВС, которые применяются для параллельно-конвейерной обработки данных, имеющих преимущественно однородный состав (массивы, матрицы, вектора, списковые наборы,

строки и др.) [3]. В составе ОВС (реконфигурируемые ВС на базе ПЛИС [4, 5] мультипроцессоры, ассоциативные процессоры [6], машины баз данных, и др.) выделяются подсистема анализа и планирования вычислений, а также операционная подсистема, представляющая программируемую вычислительную сеть функциональных блоков (узлов, ячеек) для выполнения операций над поступающими потоками однородных данных. Согласование работы двух подсистем ОВС связано с получением и обработкой унитарных кодов с помощью формул рекуррентного вида, каждый выходной бит которых описывает состояние или результат работы отдельного функционального блока (ФБ).

Область применения ОВС – решение информационных и логических поисково-переборных, вычислительно трудоемких задач, в которых выполняется ограниченный набор операций арифметического, логического, символьного характера над потоками данных однородного состава [7]. Задачи кодирования и преобразования кодовых последовательностей, задачи поиска химических и белковых сигнатур, генетического моделирования, анализа активности социальных сетей, потоковой обработки и распознавания изображений, on-line машинного перевода, математической лингвистики, комбинаторики слов [8], обработки запросов в машинах баз данных и баз знаний [9] и др. представляют такой класс задач. В них вычислительный процесс целесообразно задавать так, чтобы каждый текущий шаг вычислений учитывал свою ближайшую предысторию. Подобные вычисления свойственны модели управления потоком данных [3]. Ее главная особенность заключается в теоретически неограниченном коэффициенте ветвления вычислительных подпроцессов, особенно при обработке однородных по составу наборов данных. Используется так называемая «жадная модель параллельных вычислений» – вычисления по мере готовности операндов [3]. В соответствии с этой моделью общий вычислительный процесс представляет собой циклически взаимодействующую пару специальных процессов:

«подготовка исходных данных (операндов)» ↔
«параллельная работа функциональных блоков».

Унитарные коды (УК) являются информационной базой между двумя типами подсистем, обеспечивая подсистему анализа и планирования необходимой информации о количестве и распределении результатов работы ФБ. Несмотря на огромный прогресс пост-фоннеймановских архитектурных, структурных и схемотехнических решений [2, 10] для высокопроизводительных ВС обработка УК остается недооцененной областью, прежде всего на мезо- и микроуровнях синтеза математических моделей и проектирования конвейерных вычислительных схем.

Исходя из этого представляется востребованной задача синтеза формул рекуррентного вида для обработки УК на

основе параллельно-конвейерного подхода и разработка однородных конвейерных схем для УК.

Постановка задачи

При организации параллельных вычислений в ОВС унитарные коды играют информационно-сигнальную роль о количестве выполняемых вычислительных процессов (операций), пространственном расположении операндов в обрабатываемых информационных структурах. От скорости анализа/преобразования УК зависит работа ОВС, в целом.

Формулы рекуррентного вида являются математической формой описания преобразований над УК как битовыми строками. Соответственно, такие формулы служат основой для схем распараллеливания вычислений над УК в наиболее общем виде их представления [11, 12].

Как известно, в работе конвейера выделятся этапы загрузки и потактовой работы. При полной загрузке и подаче входных данных в виде подготовленного потока время работы конвейера будет определяться задержкой ступени с тактированием передачи данных между ступенями конвейера. Дальнейшее повышение производительности конвейера определяется увеличением количества выполняемых операций и числом одновременно работающих ступеней в единицу времени, т.е. повышением удельной производительности.

Общесистемные требования эффективной обработки УК определяют зависимость удельной производительности P обработки унитарных кодов от следующих показателей:

– длины кода (*length*);

– направления обработки бит (*direct*);

– количества выполняемых функций (*functions*);

– количество стартовых точек (*start_p*).

Соответственно целесообразны такие вычислительные схемы УК, для которых

$$P(length, direct, functions, start\_p) \rightarrow \max . \quad (1)$$

Материалы и методы

Унитарные n-разрядные коды [13, 14] представляют собой особую структуру данных, одновременно имеющую свойства числовой и символьной информации. С одной стороны, УК хранит числовой код, в котором количество логических «1» в произвольных позициях является эквивалентом числа. Важнейшая особенность УК – вариативность представления чисел и позиционная независимость элементов кода между собой, что открывает пути для распараллеливания вычислений. С другой стороны, УК – это битовая строка, отражающая своими логическими «1» структурно-лингвистические свойства или задающая отношения между отдельными подстроками. Прежде всего речь идет о расположении информационно-значимых групп бит (маркерных групп), описывающих свойства и отношения вхождения, пересечения, дополнения подстрок, предшествования или следования целевых подстрок в составе УК. Эти свойства крайне важны при об-

работке символьной информации, анализе информации в составе экспертных систем, систем поддержки принятия решений, в естественно-языковых системах, системах машинного перевода [9], а также аппаратно-программных комплексах помехоустойчивого кодирования информации, систем киберзащиты информации и др. [15].

Важнейшими операциями над УК как строками являются операции (табл. 1).

Трактовка бит в составе УК как цифр или как символов позволяет расширить возможности их параллельной обработки. Данный дуализм в трактовке структур данных (число, строка) обеспечивает выполнение числовых операций, например сравнение чисел или инкремент, на основе символьных вычислений, и наоборот – выполнение традиционных символьных операций, например арбитраж или сортировка, на основе операций цифровой обработки. В отличие от позиционного представления чисел строка отличается свойствами инвариантности обработки по направлению (слева или справа), естественной иерархической обработки по фрагментам строки, открытости к заданию нескольких стартовых точек при обработке строки и др. Эти свойства достаточно полно используются в моделях синтаксического разбора строк, в машинах вывода экспертных систем, операциях сортировки, селекции, выборки элементов и др., но непосредственно в обработке УК ограничены.

Таблица 1. Строковые операции над УК

Table 1. String operations on the UC

Строковая операция над УК / String operation on the CC	Исходный УК / Source CC	Результат / Result
Поиск первой (слева) логической «1»	0010 1010	0010 0000
Поиск последней (справа) логической «1»	0010 1010	0000 0010
Правое дополнение серии логических «1»	0010 1010	0011 1111
Левое дополнение серии логических «1»	0010 1010	1111 1110
Формирование наидлиннейшей серии логических «1»	0010 1010	0011 1110
Поиск левой наикратчайшей серии логических «1»	0010 1010	0011 1000
Поиск правой наикратчайшей серии логических «1»	0010 1010	0000 1110
Нормализация кода (левая)	0010 1010	1110 0000
Нормализация кода (правая)	0010 1010	0000 0111

Тем не менее, конвейеризация строковых операций над битами УК имеет особенности в части организации и взаимодействия множества локальных вычислительных процессов, что обусловливает разработку соответствующих математических моделей (рекуррентных формул) и согласованных с ними конвейерных схем.

В общем случае, вычислительный конвейер состоит из конечной последо-

вательности ФБ (ступеней конвейера) без циклов и соединяющих их каналов передачи данных [16]. Количество ФБ задают глубину конвейера. Каждый ФБ имеет один канал для приема данных и один канал для выдачи данных в соседнюю ступень вычислительного конвейера. Входной и выходной ФБ принимают и выдают во внешнюю среду исходные данные и результат соответственно.

Общая сумма времени выполнения операций по всем ступеням называется временем T_{DELAY} задержки (загрузки) вычислительного конвейера. Эта величина определяется как

$$T_{DELAY} = \sum_{i=1}^{i=k} t\_block_i, \qquad (2)$$

где t_block_i – время задержки i-й ступени; k – количество ступеней конвейера.

Каждая ступень вычислительного конвейера выполняет: чтение данных из своего входного канала (τ_{rd}), заданную операцию, запись результата в выходной канал (τwr) для следующей ступени. Время срабатывания i-й ступени конвейера определяется исходя из потоковой модели поступления данных в конвейер, т.е. при непрерывной подаче порций данных процессы чтения и записи в пределах ступени конвейера реализуются параллельно. Тогда время срабатывания i-й ступени выглядит как

$$t\_block_i = \tau_{fb\_i} + \max(\tau_{Rd}, \tau_{wr}), \qquad (3)$$

где τ_{fb_i} – время выполнения операции функциональным блоком на i-й ступени конвейера.

Темп выдачи результатов вычислительным конвейером определяется максимальной задержкой срабатывания ступени конвейера. При этом если ступени вычислительного конвейера имеют различную задержку, тогда для обеспечения потоковой обработки данных необходимо в ступени с меньшей задержкой дополнительно вводить буферные элементы памяти для задания единого темпа работы конвейера.

В зависимости от состава ФБ и системы синхронизации вычисленные конвейеры делятся на однородные/неоднородные, синхронные/асинхронные соответственно [17].

Далее для обработки УК будут рассматриваться однородные синхронные вычислительные конвейеры, т.е. конвейеры, имеющие единую систему синхронизации для передачи промежуточных результатов между ступенями конвейера по тактовому сигналу и однотипный состав вычислительных ячеек.

Как известно [18], конвейерные вычислительные схемы преобразования УК относятся к классу однородных комбинационных схем – итеративных сетей [19], позволяющих каждому элементу обрабатываемого кода сопоставить вычислительную ячейку. Итеративная сеть рассматривается как однородная система вычислительных ячеек с регулярными связями близкодействия для передачи информационно-управляющих сигналов между ними. Сущность работы итеративной сети заключается в организации рекуррентных вычислений по ячейкам сети, начиная со стартовой ячейки с начальным значением (стартовой точкой) с

помощью связующей функции. Эта связующая функция необходима для передачи промежуточных значений между ячейками сети. Итеративная сеть реализует рекуррентные вычисления в соответствии с заданным направлением передачи значений связующей функции. Таким образом, итеративная сеть получает исходный УК в параллельном виде, а вычисления ведутся последовательно: от ячейки к ячейке с вычислением значений связующей функции вплоть до последней (граничной) ячейки итерационной сети.

На рис. 1 показана двумерная однонаправленная итерационная сеть для обработки входных кодов $X = x_{11}x_{12}...x_{1n}$ и $Y = y_{11}y_{12}...y_{1n}$ в выходные коды $W_1 = w_{11}w_{12}...w_{1n}$ и $W_2 = w_{21}w_{22}...w_{2n}$. Каждая вычислительная ячейка Z_{ij} реализует рекуррентные функции преобразования входных бит в выходные биты с использованием связующего бита v_{i-1j-1}

$$\begin{cases} x_{i+1\,j} = \phi_1(x_{ij}, y_{ji}, v_{i-1\,j-1}) \\ y_{i+1\,j} = \phi_2(x_{ji}, y_{ij}, v_{i-1\,j-1}) \\ v_{ij} = \phi_3(x_{ij}, y_{ji}, v_{i-1\,j-1}) \end{cases} . \quad (4)$$

Начиная со стартового значения v_{00}, вычислительный процесс в итеративной сети последовательно реализуется в двух направлениях по ячейкам сети. Выходные коды $W_1 = w_{11}w_{12}...w_{1n}$ и $W_2 = w_{21}w_{22}...w_{2n}$ рекуррентно формируются за n шагов на граничных ячейках двумерной сети.

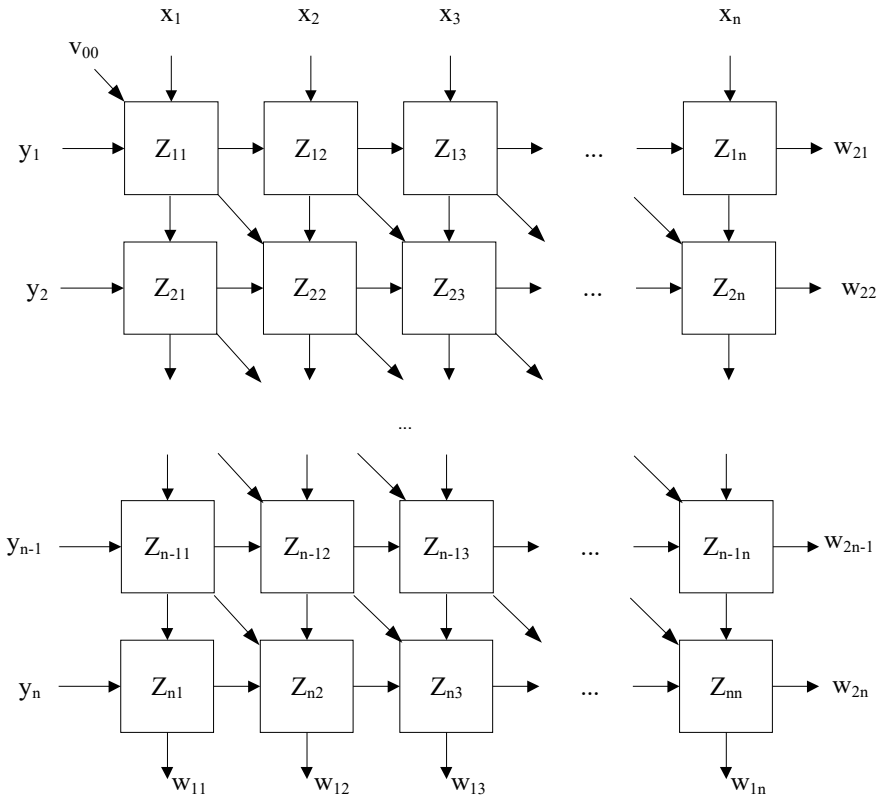

Рис. 1. Двумерная однонаправленная итерационная сеть

Fig. 1. Two-dimensional directional iterative network

Значимым прикладным примером применения двумерных итеративных сетей в вычислительных системах являются ассоциативные процессоры (АП), основу которых составляет однородный накопитель. Однородная матрица ячеек с регулярными связями в составе однородного накопителя АП используется для реализации базовых операций: поиск на совпадение/несов-падение, поиск максимального, минимального значений, всех больших или меньших значений, поиск ближайших значений и др.

На рис. 2 [20] показана структурная модель АП, реализующего одну из базовых операций, где $Я_{ij}$ –ячейка двумерной матрицы для хранения бита данных Q_{ij} и выполнения операции над ним с учетом j-го бита контекста A_j; $M1_j$ – j-ый бит маски разрядных срезов; $M2_j$ – i-ый бит маски строк однородного накопителя, РгОтв – регистр результатов (УК, АРБ – арбитр); $D_{вх}$, $D_{вых}$ – входы и выходы для подачи данных.

Ниже представлены базовые рекуррентные формулы на ij-ом шаге вычислений:

– поиск на совпадение:

$$F_{ij}^{=} = F_{ij-1}^{=} \,\&\, (\overline{M1_j} \vee \overline{A_j \oplus Q_{ij}});$$

– поиск на несовпадение

$$F_{ij}^{\neq} = F_{ij-1}^{\neq} \vee M1_j (A_j \oplus Q_{ij});$$

– поиск максимальных значений

$$F_{ij}^{\max} = F_{ij-1}^{\max} \,\&\, (\overline{M}1_{j\,j} \vee S_j \vee Q_{ij});$$

– поиск минимальных значений

$$F_{ij}^{\min} = F_{ij-1}^{\min} \,\&\, (M1_j \vee S_j \vee \overline{Q}_{ij});$$

– поиск больших/меньших

$$\begin{cases} F1_{ij} = F1_{ij-1} \,\&\, (\overline{M}1_j \vee \overline{A}_j \vee Q_{ij}) \\ F2_{ij} = F2_{ij-1} \vee F1_{ij-1} (\overline{M}1_j \, \overline{A}_j Q_{ij}) \end{cases}.$$

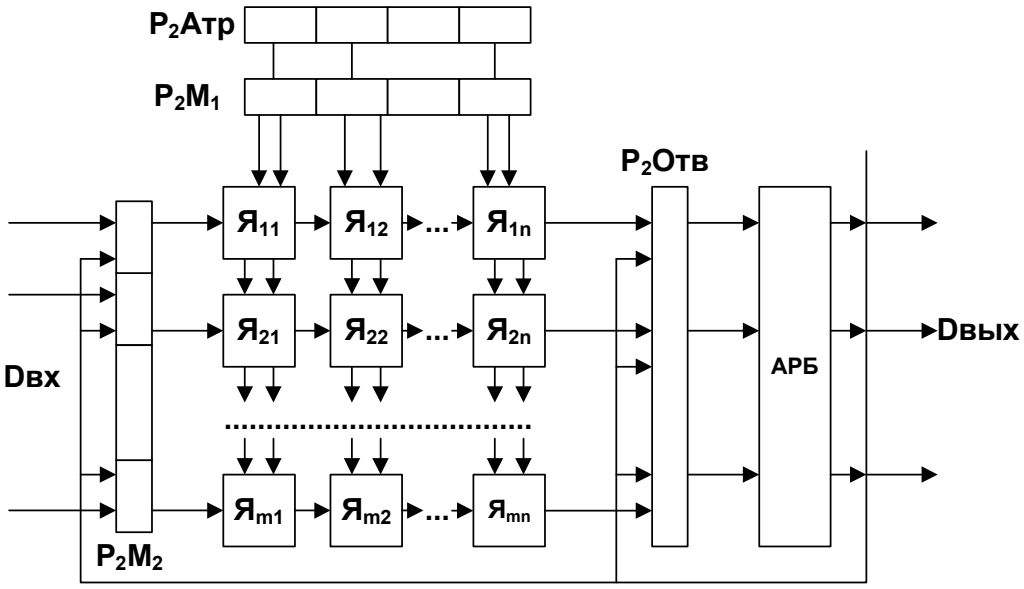

Рис. 2. Структурная модель АП с однородным накопителем

Fig. 2. Structural model of CAM

Данные операции выполняются параллельно по разрядным срезам однородного накопителя при условии размещения всех операндов в соответствующих переменных. Отличительная особенность рекуррентных формул для операций АП – задание единственной стартовой точки по границе однородного накопителя, как правило – по левой границе накопителя. Данный порядок распространения поисковых значений в АП наследован от принципа позиционных зависимости разряда числа от его позиции в числе. Строковая интерпретация данных в однородном накопителе позволяет реализовывать поиск как от левой, так и от правой границ накопителя, но эта возможность не реализована в АП.

Для обработки УК наибольшее распространение получили одномерные итеративные сети с одним направлением передачи связей между ячейками, и как правило, с единственной стартовой точкой. На рис. 3 показан общий вид одномерной итеративной сети, перерабатывающей входной код $X = x_1 x_2 ... x_n$ в выходной код $Y = y_1 y_2 ... y_n$ с заданным направлением связи ячеек слева направо с помощью связующей функции $V = v_1 v_2 ... v_n v_{n+1}$.

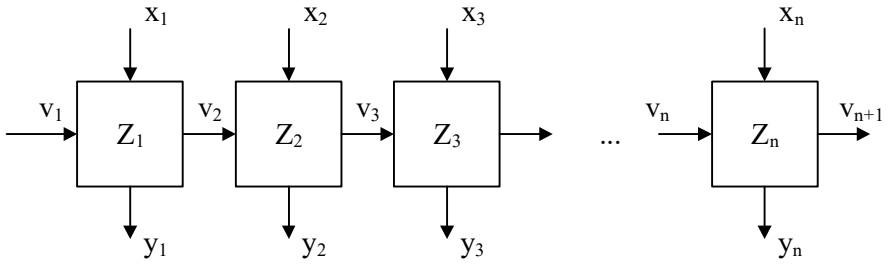

Рис. 3. Одномерная однонаправленная итерационная сеть

Fig. 3. One-dimensional directional iterative network

Здесь каждая вычислительная ячейка рекуррентно реализует связующую и выходную функции в соответствии с заданным направлением (i=1…n)

$$\begin{cases} v_{i+1} = \gamma_1(x_i, v_i) \\ y_i = \gamma_2(x_i, v_i)\,. \end{cases} \quad (5)$$

Время работы итеративной сети линейно зависит от длины кода n и составляет $T = nt_{CELL}$, где t_{CELL} – задержка одной ячейки. Динамика получения выходных бит в выходном коде $Y = y_1 y_2 ... y_n$ описывается дискретными моментами времени t_{CELL}, $2t_{CELL}$,…nt_{CELL}. Следовательно, общее время работы итеративной сети определяется моментом времени срабатывания граничной ячейки Z_n.

Ярким примером одномерной итеративной сети с одним направлением распространения вычислительного процесса является параллельный сумматор с последовательными переносами, который за n шагов вычисляет сумму двух n-разрядных операндов (входных кодов) [21].

Ниже для *n*-разрядного сумматора представлены рекуррентные формулы вычисления бита суммы s_i и бита переноса p_i в зависимости от входных бит исходных операндов a_i, b_i и входного переноса p_{i-1}

$$\begin{cases} s_i = a_i \oplus b_i \oplus p_{i-1} \\ p_i = p_{i-1}(a_i \vee b_i) \vee a_i b_i . \end{cases} \quad (6)$$

Другими известными примерами итеративных сетей являются схемы арбитров, поисковые или кодирующие схемы [22], трактующие входной код как строку с их последовательным поэлементным преобразованием.

Тем не менее, последовательный характер связей между вычислительными ячейками, определённый рекуррентным видом преобразующих функций вида (1), (2), определяет линейную зависимость времени T вычислений от длины n входного УК. Главная причина – единственный вычисленный процесс, начинающийся со стартовой точки и охватывающий все вычислительные ячейки итерационной сети.

Для повышения производительности итерацитивные сети организуются как двунаправленные, используя с двух сторон связующие коды $V1 = v_{11}v_{12}...v_{1n}$ $V2 = v_{21}v_{22}...v_{2n}$ и задавая в них две самостоятельные стартовые точки (v_{11}, v_{2n}) для двух встречных вычислительных процессов (рис. 4), при условии локализации вычислительной операции над УК по частям.

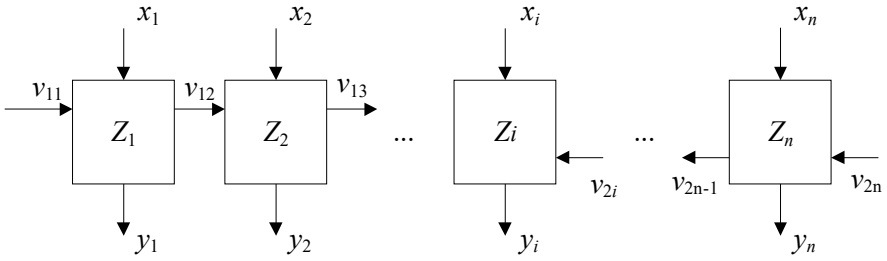

Рис. 4. Одномерная двунаправленная итерационная сеть

Fig. 4. One-dimensional bidirectional iterative network

При такой организации итеративной сети итоговый выходной код $Y = y_1 y_2 ... y_n$ по всем разрядам будет формироваться с двух направлений за время, равное половине длины n УК, т.е. за $T = (n/2)t_{CELL}$. В зависимости от свойства четности/нечетности длины n УК последней будет вычисляться или центральная ячейка итеративной сети Z_i или вычисления завершатся одновременно по граничным ячейкам двух подстрок длиной $n/2$.

Для АП с операцией «поиск на совпадение» вычисления двумя встречными процессами описываются следующей системой рекуррентных формул:

$$\begin{cases} F1_{iq1}^{=} = F1_{iq1-1}^{=}(\overline{M1_{q1}} \vee \overline{A_{q1} \oplus Q_{q1}}), q1 = 1...n/2. \\ F2_{iq2}^{=} = F2_{iq2-1}^{=}(\overline{M1_{q2}} \vee \overline{A_{q2} \oplus Q_{q2}}), q2 = n...(n/2+1) \\ F_i^{=} F1_{i(n/2)}^{=} \& F2_{i(n/2+1)}^{=}. \end{cases} \quad (7)$$

При симметрии вычислительных процессов (большинство строковых операций над УК) связующие коды $V1$, $V2$ будут иметь совпадающую обратно симметрич-

ную структуру. Также будут равны стартовые значения с обеих граничных ячеек итерационной сети - Z_1 и Z_n. Тогда выражение (2) для раздельных индексных переменных $q1=(1\ldots n/2)$ и $q2=(n\ldots n/2+1)$ уточняется как

$$\begin{cases} v_{1q1+1} = \gamma_1(x_{q1}, v_{1q1}) \\ y_{q1} = \gamma_2(x_{q1}, v_{1q1}) \\ v_{2q2} = \gamma_1(x_{q2}, v_{2q2+1}) \\ y_{q2} = \gamma_2(x_{q2}, v_{2q2}) \end{cases} . \quad (8)$$

Для двунаправленной итеративной сети 2 текущие ячейки Z_{q1}, Z_{q2}, ($q1=1\ldots n/2$, $q2=n\ldots n/2+1$) формируют 2 бита v_{q1+1}, v_{q2-1} связующих кодов для работы двух следующих вычислительных ячеек Z_{q1+1}, Z_{q2-1}, также вычисляются два выходных бита в составе $Y = y_1 y_2 \ldots y_n$.

Вместе с тем организация двух встречных вычислительных процессов с собственными стартовыми точками не влечет повышение удельной производительности итеративной сети, так как каждая ячейка выполняет лишь одно преобразование. Более перспективным является мультифункциональный подход к обработке УК, т.е. одновременного выполнения итеративной сетью не менее 2-х функций.

На рис. 5 показана организация мультифункциональной итеративной сети с двумя встречными вычислительными процессами, при этом время формирования двух выходных кодов $Y1 = y_{11} y_{12} \ldots y_{1n}$ $Y2 = y_{21} y_{22} \ldots y_{2n}$ составит $T = (n/2) t_{CELL}$ за счет того, что в каждый момент времени в ячейках сети будут реализовываться не менее 2-х рекуррентных формул обработки бит входного УК – $X = x_1 x_2 \ldots x_n$.

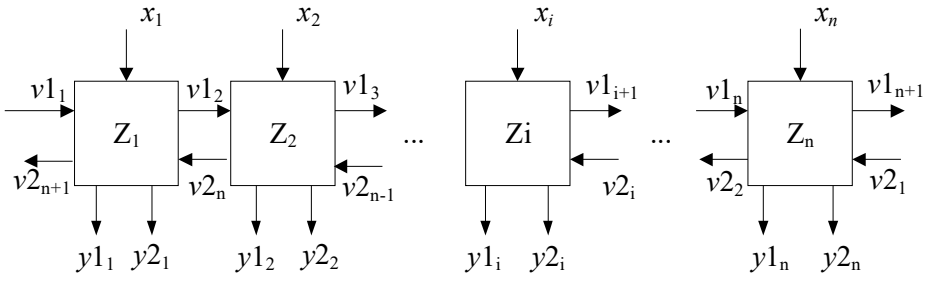

Рис. 5. Двунаправленная итеративная сеть

Fig. 5. Bidirectional iterative network

Следующий путь повышения производительности – организация параллельно-конвейерных вычислений путем специальной подготовки потока входных кодов для совмещения во времени обработки элементов УК из разных наборов.

Тактирование работы ячеек итерационной сети позволит выдавать текущие результаты на каждом такте работы конвейера, что приводит к исключению потерь времени на ожидание следующего элемента УК в потоке.

В зависимости от задачи обработки УК на рис. 6 показаны схемы подготовки наборов УК для мультифункциональной обработки, начиная вычислительные процессы с граничных ячеек итерационной или от центральной ячейки сети.

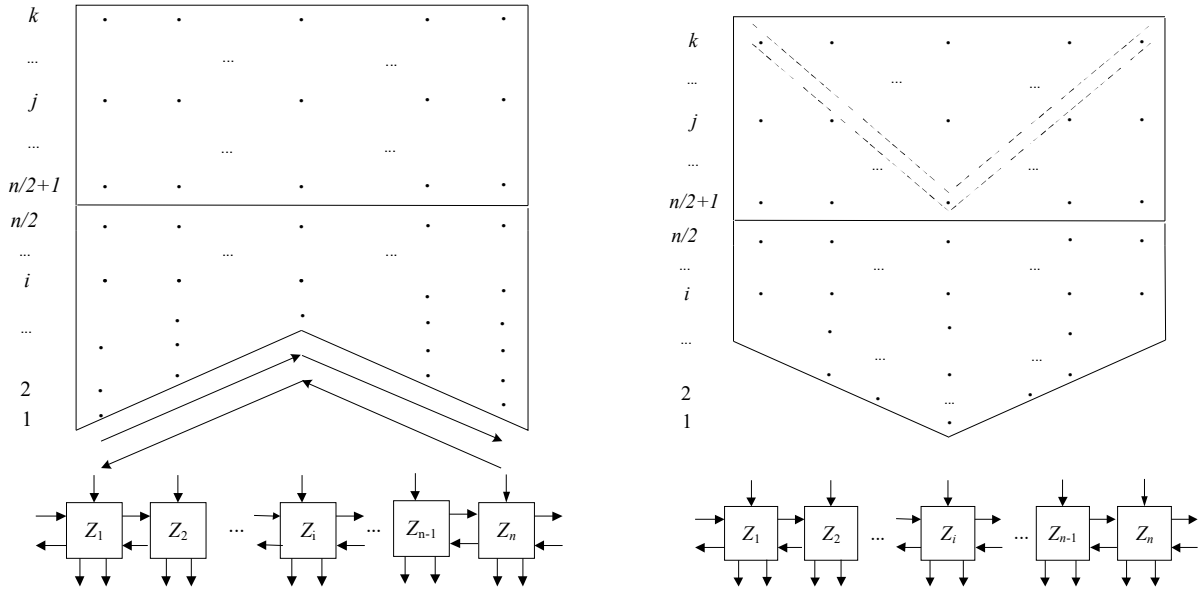

Рис. 6. Подготовка данных и мультифункциональная обработка УК

Fig. 6. Data preparation and multifunctional processing of the UC

При мультифункциональной организации конвейерных схем (рис. 6) сокращается время загрузки конвейера, в дальнейшем осуществляется совмещение и обработка элементов различных УК с выдачей на каждом такте не менее двух выходных бит.

Повышение удельной производительности конвейерного выполнения одной операции на итеративной сети достигается за счет предварительного вычисления стартовых точек и организации множества вычислительных процессов (встречных или сонаправленных) по длине *n* УК.

Такое распараллеливание вычислений основано на дуальной трактовке УК как строки при выполнении набора операций (см. табл. 1).

Мультиконвейерная обработка сводится к формированию локальных конвейеров, работающих одновременно по длине обрабатываемых унитарных кодов и имеющих для этого собственные стартовые точки. Эта возможность достигается за счет получения дополнительной информации системного характера о состоянии УК, позволяющей назначать стартовые точки исходя из свойств отдельных фрагментов УК (префикс, тело, суффикс).

Для мультиконвейерной обработки над фрагментами УК выполняются многовходовые операции ИЛИ, И, ИЛИ-НЕ, И-НЕ для выявления информационно-значимых ситуаций для поиска логических «1» или «0», сброса логических «1» по длине кода, выявления кодовых сигнатур 11..1, 00..0 и др.

Каждый локальный конвейер в составе мультиконвейера будет иметь собственную стартовую точку, что позволит всем локальным конвейерам в составе итерационной сети вести параллельную работу.

На рис. 7 показаны схемы организации локальных конвейеров в составе итерационной сети.

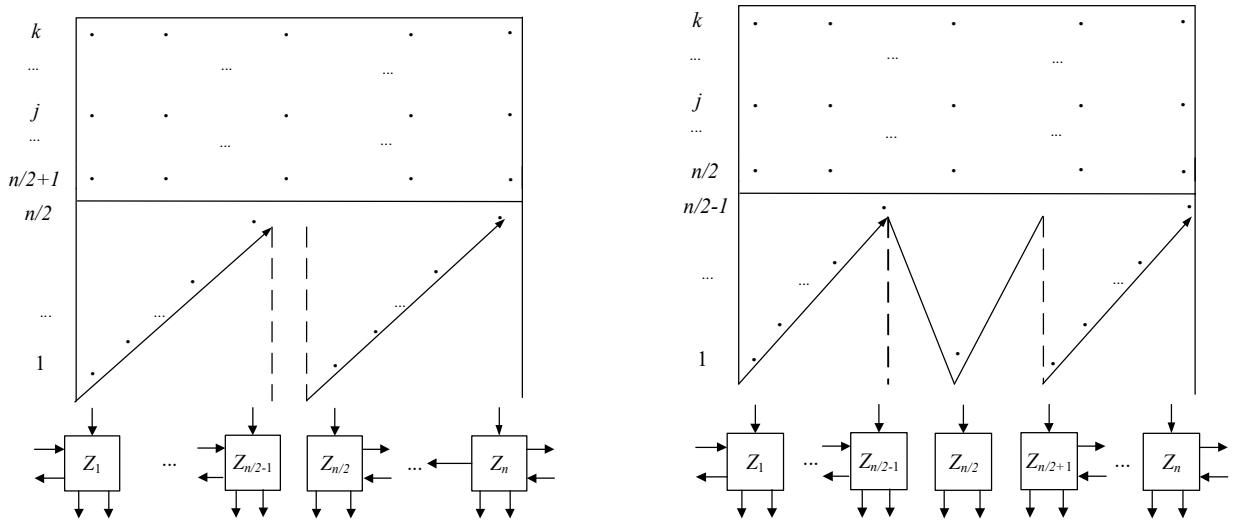

Рис. 7. Организация мультиконвейерной обработки

Fig. 7. Organization of multi-pipeline processing

Таким образом, сокращение времени работы конвейерных схем для обработки УК достигается на основе следующих принципов: тактирование, мультифункциональность ячеек конвейера, мультиконвейеризация, т.е. декомпозиция единого конвейера на множество локальных конвейеров с собственными стартовыми точками. Данные принципы позволяют использовать имеющуюся теоретическую базу обработки УК в виде рекуррентных формул с учетом введения раздельного контроля границ для индексных переменных по выполняемым операциям и назначения стартовых точек для локальных конвейеров, работающих одновременно в составе итеративной сети.

Результаты и их обсуждение

С практической точки зрения наиболее востребованными в аппаратных способах, схемах и устройствах являются следующие операции над УК:

– поиск первой (слева) логической «1», операция $Y1$;

– формирование правой серии логических «1», операция $Y2$;

– поиск последней (справа) логической «1», операция $Y3$;

– формирование левой серии логических «1», операция $Y4$.

Используя принципы мультифункциональности, мультиконвейеризации, рассматривается схема конвейера для выполнения операций $Y1$-$Y4$ в рамках одной итерационной сети (рис. 8).

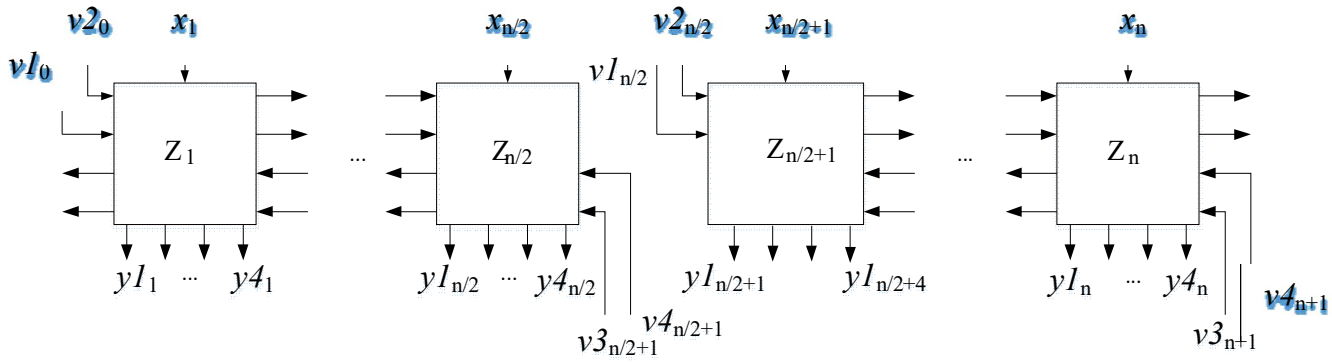

Рис. 8. Конвейерная схема для реализации операций Y1-Y4

Fig. 8. Conveyor scheme for implementing operations Y1-Y4

На рис. 9 показана i-я ячейка итеративной сети для вычисления двух бит $y1_i$, $y2_i$ операций поиска первой (левой) логической «1» и формирования правой серии логических «1». Направление обхода ячеек сети для операций $Y1$, $Y2$ – от левой границы к правой границе сети.

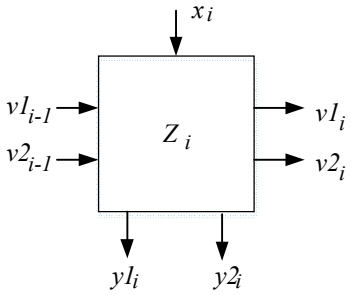

Рис. 9. Ячейка итеративной сети для операций Y1, Y2

Fig. 9. Iterative network cell for operations Y1, Y2

Стартовые точки $v1_0$,$v2_0$ связующих функций имеют по умолчанию значения $v1_0$=0 и $v2_0$=0. Они кодируют начальные состояния поиска первой (левой) логической «1» и формирования (генерации) серии логических «1» от найденной логической «1» соответственно.

Стартовые точки $v1_{n/2}$,$v2_{n/2}$ связующих функций получают начальные значения на основе многовходового элемента ИЛИ от элементов левой половины УК. Действительно, если левая половина УК уже содержит хотя бы одну логическую «1», то второй локальный конвейер должен вести сброс всех логических «1» во второй половине УК (для операции $Y1$) и генерацию логических «1» во всех позициях элементов второй половины (для операции $Y2$). В таблице 2 приведены таблицы истинности для i-го бита операций $Y1$, $Y2$.

Таблица 2. Таблица истинности работы ячейки на операциях Y1, Y2

Table 2. Table of cell operation on operations Y1, Y2

$v1_{i-1}$	x_i	$y1_i$	$v1_i$
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	1

$v2_{i-1}$	x_i	$y2_i$	$v2_i$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Аналогичным образом создается ячейка и таблицы истинности для вычисления i-ых выходных бит операций $Y3$, $Y4$ с применением связующих бит $v3_i$, $v4_i$ для них. Направление обхода ячеек сети для операций $Y3$, $Y4$ – от левой границы к правой границе сети (табл. 3).

Рекуррентные формулы для реализации данных операций на мультиконвейере имеют вид

$$\begin{cases} v1_{q1} = x_{q1} \vee v1_{q1-1}, q1 = 1..n/2, v1_0 = 0 \\ y1_{q1} = x_{q1} \,\&\, \overline{v}1_{q1-1}, q1 = 1..n/2 \\ v2_{q1} = x_{q1} \vee v2_{q1-1}, q1 = 1..n/2, v2_0 = 0 \\ y2_{q1} = x_{q1} \vee v2_{q1-1}, q1 = 1..n/2, \end{cases} \quad (9)$$

$$\begin{cases} v1_{q2} = x_{q2} \vee v1_{q2-1}, q2 = n/2+1..n, v1_{n/2} = OR_1^{n/2}(x_k) \\ y1_{q2} = x_{q2} \,\&\, \overline{v}1_{q2-1}, q2 = n/2+1..n, \\ v2_{q2} = x_{q2} \vee v2_{q2-1}, q2 = n/2+1..n, v2_{n/2} = OR_1^{n/2}(x_k) \\ y2_{q2} = x_{q2} \vee v2_{q2-1}, q2 = n/2+1..n, \end{cases} \quad (10)$$

$$\begin{cases} v3_{q2} = x_{q2} \vee v3_{q2+1}, q2 = n..n/2+1, v3_{n+1} = 0 \\ y3_{q2} = x_{q2} \,\&\, \overline{v}3_{q2+1}, q2 = n..n/2+1 \\ v4_{q2} = x_{q2} \vee v4_{q2+1}, q2 = n..n/2+1, v4_{n+1} = 0 \\ y4_{q2} = x_{q2} \vee v4_{q2+1}, q2 = n..n/2+1, \end{cases} \quad (11)$$

$$\begin{cases} v3_{q1} = x_{q1} \vee v3_{q1+1}, q1 = 1..n/2, v3_{n/2+1} = OR_{n/2+1}^{n}(x_k) \\ y3_{q1} = x_{q1} \,\&\, \overline{v}3_{q1+1}, q1 = 1..n/2 \\ v4_{q1} = x_{q1} \vee v\quad 4_{q1+1}, q1 = 1..n/2, v4_{n/2+1} = OR_1^{n/2}(x_k) \\ y4_{q1} = x_{q1} \vee v4_{q1+1}, q1 = 1..n/2. \end{cases} \quad (12)$$

Таблица 3. Таблица истинности работы ячейки на операциях $Y3$, $Y4$

Table 3. Table of cell operation on operations Y3, Y4

$v3_i$	x_i	$y3_i$	$v3_{i-1}$
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	1

$v4_i$	x_i	$y4_i$	$v4_{i-1}$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Оценка временной задержки ступени t_{level} конвейера содержит время срабатывания ячейки t_{lcell} сети и время тактирования (задержка, например, D-триггера), необходимое для сохранения и передачи промежуточных данных между ступенями. Также учитывается время подготовки данных для подачи на ячейки конвейера. С учетом выражений (9)-(12) $t_{level}=4\tau$, где τ – время логического вентиля. Тогда эффект от мультифункциональности и мультиконвейеризации оценивается временем, приходящимся на одну операцию.

На рис. 10 показаны времена (в усл. единицах) на одну операцию для моно- и мультиконвейеров, где параметры конвейера: $m1$ – количество функций; $m2$ – количество локальных конвейеров.

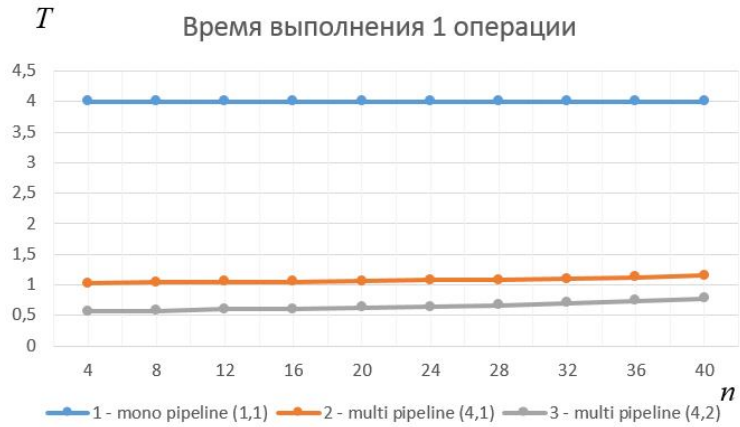

Рис. 10. Время, приходящееся на одну операцию в моно- и мультиконвейерах

Fig. 10. Time per operation in mono- and multi-pipelines

Данные графики показывают повышение удельной производительности конвейерных схем на базе двунаправленной итеративной сети за счет параллельного вычисления нескольких функций в составе ячеек сети и специально организованных локальных конвейерах.

Выводы

Эффективная работа современных однородных ВС поддерживается параллельно-конвейерной обработкой унитарных кодов.

Унитарные коды широко используются во многих проблемно-поисковых и информационно-аналитических, задачах обработки изображений, когнитивных задачах и процессах при планировании параллельных вычислений [23]. Для их эффективной обработки целесообразно использовать многофункциональные конвейеры с параллельной работой локальных конвейеров, имеющих собственные стартовые точки. Объединение принципов тактирования ячеек, выбора направления обработки унитарных кодов, мультифункциональности ячеек, мультиконвейеризации позволяет повысить удельную производительность конвейера при соответствующем увеличении аппаратных затрат на специальную подготовку и подачу элементов кодов из входных потоков. Оценка времени, приходящегося на одну выполняемую операцию, показала не менее 4-кратного временного выигрыша.

Список литературы

1. Корнеев В. В. Вычислительные системы. М.: Гелиос АРВ, 2004. 510 с.

2. Хорошевский В. Г. Архитектура вычислительных систем. 2-е изд. М.: Московский государственный технический университет им. Н.Э. Баумана, 2008. 520 с.

3. Бурцев В. С. Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ: сб. статей. М.: Торус Пресс, 2006. 414 с.

4. Гузик В.Ф., Каляев И.А., Левин И.И. Реконфигурируемые вычислительные системы. Ростов-на-Дону: Южный федеральный университет, 2016. 472 с.

5. Каляев И. А., Левин И.И. Реконфигурируемые вычислительные системы на основе ПЛИС. Ростов-на-Дону: Южный научный центр РАН, 2022. 475 с.

6. Огнев И.В., Борисов В.В., Сутула Н.А. Ассоциативные память, среды, системы. М.: Горячая линия – Телеком, 2016. 420 с.

7. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.

8. Lothaire M. Applied Combinatorics on Words // Encyclopedia of Mathematics and its Applications. Cambridge: Cambridge University Press, 2005.

9. Рыбина Г.В. Основы построения интеллектуальных систем. М.: Финансы и статистика, ИНФРА-М, 2010. 432 с.

10. Архитектура вычислительных систем с элементами конвейерной обработки / О.Я. Кравец, Е.С. Подвальный, В.С. Титов, А.С. Ястребов. Воронеж, Курск, Санкт-Петебург: Политехника, 2009. 151 с.

11. Воеводин В.В. Математические модели и методы в параллельных процессах. М.: Наука, 1986. 296 с.

12. Voevodin V.V., Voevodin Vl.V. Parallel computing. St. Petersburg: BHV-Peterburg Publ., 2002. 608 p.

13. Бандман О.Л., Миренков Н.Н., Седухин С.Г. Специализированные процессоры для высоко-производительной обработки данных. М.: Радио и связь, 1988. 208 с.

14. Фет Я. И. Параллельные процессоры для управляющих систем. М.: Энергоиздат, 1981. 160 с.

15. Обзор исследований киберстрахования / А. Л. Ханис, С. В. Беспалько, Е. А. Титенко [и др.] // Интеллектуальные информационные системы: тенденции, проблемы, перспективы: материалы докладов VII всероссийской очной научно-практической конференции «ИИС-2019». Курск: Юго-Западный государственный университет, 2019. С. 108-118.

16. Павлов П. А. Математические модели и методы организации вычислений в мультипроцессорных системах // Компьютерные исследования и моделирование. 2025. Т. 17, № 3. С. 423-436.

17. Кобайло А. С. Применение гибридных методов проектирования вычислительных систем реального времени // Труды БГТУ. Серия 3: Физико-математические науки и информатика. 2018. № 1(206). С. 120-124.

18. Ватутин Э. И., Зотов И.В., Титов В.С. Использование схемных формирователей и преобразователей двоичных последовательностей при построении комбинатор-

но-логических акселераторов // Известия Курского государственного технического университета. 2008. №4(25). С. 32-39.

19. Зельдин Е.А. Цифровые интегральные микросхемы в информационно-измерительной аппаратуре. Л.: Энергоатомиздат, 1986. 280 с.

20. Аппаратно-ориентированный метод ускоренного поиска вхождений образца на основе структурно-процедурных вычислений / Е. А. Титенко, Э. И. Ватутин, М. А. Титенко [и др.] // Известия ЮФУ. Технические науки. 2024. № 5(241). С. 68-78.

21. Титенко Е. А., Скорняков К.С., Бусыгин К.Н. Методы и сумматоры с параллельными групповыми процессами // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2013. № 1. С. 161-166.

22. Титенко Е. А., Семенихин Е.А., Петрик Е. А. Структурно-функциональная организация арбитра параллельной обработки запросов // Информационно-измерительные и управляющие системы. 2010. Т. 8, № 11. С. 30-34.

23. Титенко Е. А., Типикин А. П., Лапин Д. В. Некоторые пути построения перспективных вычислительных систем для параллельной обработки массивов данных и изображений на ПЛИС // Электромагнитные волны и электронные системы. 2016. Т. 21, № 10. С. 56-59.

References

1. Korneev V. V. Computing systems. Moscow: Gelios ARV; 2004. 510 p. (In Russ.).

2. Khoroshevsky V. G. Architecture of computing systems. 2nd ed. Moscow: Moskovskii gosudarstvennyi tekhnicheskii universitet im. N.E. Baumana; 2008. 520 p. (In Russ.).

3. Burtsev V. S. Parallelism of computing processes and development of supercomputer architecture. Collection of articles. Moscow: TORUS PRESS; 2006. 414 p. (In Russ.).

4. Guzik V. F., Kalyaev I. A., Levin I. I. Reconfigurable computing systems Rostov-on-Don: Yuzhnyi federal'nyi universitety; 2016. 472 p. (In Russ.).

5. Kalyaev I. A., Levin I. I. Reconfigurable computing systems based on FPGAs. Rostov-on-Don:Yuzhnyi federal'nyi universitet; 2022. 475 p. (In Russ.).

6. Ognev I. V., Borisov V. V., Sutula N. A. Associative memory, environments, systems Moscow: Goryachaya Liniya – Telecom; 2016. 420 p. (In Russ.).

7. Gary M., Johnson D. Computing machines and difficult-to-solve problems. Moscow: Mir; 1982. 416 p. (In Russ.).

8. Lothaire M. Applied Combinatorics on Words. In: *Encyclopedia of Mathematics and its Applications*. Cambridge: Cambridge University Press; 2005.

9. Rybina G. V. Fundamentals of Building Intelligent Systems. Moscow: Finansy i statistika; INFRA-M; 2010. 432 p. (In Russ.).

10. Kravets O. Ya., Podvalny E. S., Titov V. S., Yastrebov A. S. Architecture of Computing Systems with Elements of Conveyor Processing. Voronezh, Kursk, Sankt-Peteburg: Politekhnika; 2009. 151 p. (In Russ.).

11. Voevodin V. V. Mathematical Models and Methods in Parallel Processes. Moscow: Nauka; 1986. 296 p. (In Russ.).

12. Voevodin V. V., Voevodin Vl. V. Parallel Computing. St. Petersburg: BHV-Peterburg Publ.; 2002. 608 p. (In Russ.).

13. Bandman O.L., Mirenkov N.N., Sedukhin S.G. Specialized processors for high-performance data processing. Moscow: Radio i svyaz; 1988. 208 p. (In Russ.).

14. Fet Ya. I. Parallel processors for control systems. Moscow: Energoizdat; 1981. 160 p. (In Russ.).

15. Khanis A. L., Bespalko S. V., Titenko E. A., et al. Review of cyber insurance research. In: *Intellektual'nye informatsionnye sistemy: tendentsii, problemy, perspektivy: materialy dokladov VII vserossiiskoi ochnoi nauchno-prakticheskoi konferentsii «IIS-2019» = Intelligent information systems: trends, problems, prospects. Materials of reports of the VII All-Russian face-to-face scientific and practical conference "IIS-2019".* Kursk: Southwest State University; 2019. P. 108-118. (In Russ.).

16. Pavlov P. A. Mathematical models and methods for organizing computations in multiprocessor systems. *Komp'yuternye issledovaniya i modelirovanie = Computer research and modeling*. 2025; 17(3): 423-436. (In Russ.).

17. Kobailo A. S. Application of hybrid methods for designing real-time computing systems. *Trudy BGTU. Seriya 3: Fiziko-matematicheskie nauki i informatika = Proceedings of BSTU. Series 3: Physical and mathematical sciences and computer science*. 2018; (1): 120-124. (In Russ.).

18. Vatutin E. I., Zotov I. V., Titov V. S. Use of circuit shapers and binary sequence converters in the construction of combinatorial logic accelerators. *Izvestiya Kurskogo gosudarstvennogo tekhnicheskogo universiteta = Proceedings of Kursk State Technical University*. 2008; (4): 32-39. (In Russ.).

19. Zel'din E.A. Digital integrated circuits in information-measuring equipment. Leningrad: Energoatomizdat; 1986. 280 p. (In Russ.).

20. Titenko E. A., Vatutin E. I., Titenko M. A., et al. Hardware-oriented method for accelerated search for sample occurrences based on structural-procedural calculations. *Izvestiya YuFU. Tekhnicheskie nauki = Bulletin of the Southern Federal University. Technical sciences*. 2024; (5): 68-78. (In Russ.).

21. Titenko E. A., Skornyakov K.S., Busygin K.N. Methods and adders with parallel group processes. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta. Seriya: Upravlenie, vychislitel'naya tekhnika, informatika. Meditsinskoe priborostroenie = Proceedings of*

the Southwest State University. Series: Control, Computing Engineering, Information Science. Medical Instruments Engineering. 2013; (1): 161-166. (In Russ.).

22. Titenko E. A, Semenikhin E. A., Petrik E. A. Structural and functional organization of the arbitrator of parallel query processing. *Informatsionno-izmeritel'nye i upravlyayushchie sistemy = Information, measuring and control systems*. 2010; 8(11): 30-34. (In Russ.).

23. Titenko E. A., Tipikin A. P., Lapin D. V. Some ways of constructing promising computing systems for parallel processing of data arrays and images on FPGA. *Elektromagnitnye volny i elektronnye sistemy = Electromagnetic waves and electronic systems*. 2016; 21(10): 56-59. (In Russ.).

Информация об авторах / Information about the Authors

Титенко Евгений Анатольевич, кандидат технических наук, доцент, доцент кафедры программной инженерии, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: johntit@mail.ru, ORCID: http://orcid.org/0000-0002-5659-4747

Evgeny A. Titenko, Cand. of Sci. (Engineering), Associate Professor, Associate Professor of the Software Engineering Department, Southwest State University, Kursk, Russian Federation, e-mail: johntit@mail.ru, ORCID: http://orcid.org/0000-0002-5659-4747

Сизов Александр Семенович, доктор технических наук, профессор, профессор кафедры программной инженерии, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: johntit@mail.ru, ORCID: http://orcid.org/0000-0001-8110-9929

Alexandr S. Sizov, Dr. of Sci. (Engineering), Professor, Professor of the Software Engineering Department, Southwest State University, Kursk, Russian Federation, e-mail: johntit@mail.ru, ORCID: http://orcid.org/0000-0001-8110-9929

Титенко Михаил Андреевич, аспирант кафедры программной инженерии, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: johntit@mail.ru, ORCID: http://orcid.org/0009-0008-1560-5594

Mikhail A. Titenko, Post-Graduate Student, Southwest State University, Software Engineering Department, Kursk, Russian Federation, e-mail: johntit@mail.ru, ORCID: http://orcid.org/0009-0008-1560-5594