noyabrya 2018 g. [Certificate of state registration of computer software № 2018664980. Calculation of probabilities of maximum frequency statistics. Proprietor – A.K. Melnikov. Authors: A.K. Melnikov and N.B. Zeliukin. Application № 2018662123. Date of filing – November 01, 2018. Date of state registration in the Register of computer software – November 27, 2018].
19. *Mel'nikov A.K.* Metodika rascheta raspredeleniya veroyatnostey znacheniy simmetrichnykh additivno razdelyaemykh statistik, priblizhennykh k ikh tochnomu raspredeleniyu [A method for calculating the probability distribution of values of symmetric additively separated statistics that are close to their exact distribution], *Nauchnyy vestnik NGTU* [Science bulle-tin of the Novosibirsk state technical university], 2018, No. 1 (70), pp. 153-166. ISBN 1814-1196. Doi: 10.17212/1814-1196-2018-1-153-166.
20. *Hutchinson T.P.* 1979. The validity of the chi-squared test when expected frequencis are small: A list of recent research references, *Commun. Stat. A Theor*., Vol. 8, No. 4, pp. 327-335.

Статью рекомендовал к опубликованию д.т.н., профессор И.И. Левин.

Мельников Андрей Кимович – АО «Вычислительные решения»; e-mail: ak@comp-sol.ru; 117587, г. Москва, Варшавское шоссе, 125, тел.: +74952870035; к.т.н.; доцент ВАК; г.н.с.

Melnikov Andrey Kimovich – SC «Computing solutions»; e-mail: ak@comp-sol.ru; 125, Varshavskoye roag, Moscow, 117587, Russia; phone: +784952870035; cand. of eng. sc.; associate professor of SAC; chief research officer.

УДК 519.178 DOI 10.18522/2311-3103-2020-7-78-93

Д.В. Михайлов

ПРЕОБРАЗОВАНИЕ НЕКОТОРЫХ ВИДОВ ПОСЛЕДОВАТЕЛЬНЫХ ИНФОРМАЦИОННЫХ ГРАФОВ В ПАРАЛЛЕЛЬНО-КОНВЕЙЕРНУЮ ФОРМУ

Многие задачи цифровой обработки сигналов могут быть представлены в виде информационных графов. Реконфигурируемые вычислительные системы, построенные на основе ПЛИС, могут иметь структуру, непосредственно соответствующую информационному графу решаемой задачи. Построение графа задачи и последующее создание вычислительной структуры может занимать значительное время при выполнении их вручную. В связи с этим возникает необходимость создания алгоритмов преобразования информационных графов, которые могут выполняться автоматически. В статье предложены алгоритмы преобразования однородных графов, содержащих ассоциативные операции, и смешанных графов, содержащих два типа операций, один из которых является дистрибутивным по отношению к другому. Преобразования графов первого типа (состоящих из операций одного типа) сводятся к переходу от последовательной формы графа к пирамидальной для ускорения выполнения всех операций графа. В случае если имеющегося количества оборудования недостаточно для реализации всех операций графа, применяется преобразование, разбивающее исходный граф на изоморфные подграфы. Размер подграфа зависит от имеющегося вычислительного ресурса. В этом случае вычислительная структура будет соответствовать такому подграфу. Преобразования графов второго типа (состоящих из операций двух типов, одни из которых являются дистрибутивными по отношению к другим) сводятся к разделению графа на подграфы, содержащие операции одного типа, соединённые особым образом. После этого эти подграфы могут быть преобразованы в пирамидальную форму для ускорения выполнения всех операций графа. При этом количество вершин с дистрибутивными операциями может значительно возрасти, в связи с чем может потребоваться сокращение их числа. Отсюда следует, что при преобразовании графов второго типа не обходимо выбирать конкретную форму, к которой будет приведён граф, исходя из соотношения его размера и имеющегося вычислительного ресурса. Таким образом, предложенные алгоритмы преобразования информационных графов различных типов могут быть эффективно использованы при разработке вычислительных структур, основанных на ПЛИС.

Графы; реконфигурируемые вычислительные системы; преобразование графов.

D.V. Mikhailov

CONVERTING SOME TYPES OF SEQUENTIAL INFORMATION GRAPHS INTO PARALLEL-PIPELINE FORM

Many digital signal processing tasks can be represented in the form of information graphs. Reconfigurable computing systems based on FPGAs can have a structure that directly corresponds to the information graph of the problem being solved. The construction of the task graph and the subsequent creation of the computational structure can take a significant amount of time when performed manually. In this regard, it becomes necessary to create algorithms for transforming information graphs that can be performed automatically. The article proposes algorithms for transforming homogeneous graphs containing associative operations and mixed graphs containing two types of operations, one of which is distributive with respect to the other. Transformations of graphs of the first type (consisting of operations of the same type) are reduced to the transition from a sequential form of a graph to a pyramidal form to speed up the execution of all graph operations. If the available amount of equipment is not enough to implement all operations of the graph, a transformation is applied that splits the original graph into isomorphic subgraphs. The size of the subgraph depends on the available computing resources. In this case, the computational structure will correspond to such a subgraph. Transformations of graphs of the second type (consisting of operations of two types, some of which are distributive with respect to others) are reduced to dividing the graph into subgraphs containing operations of the same type, connected in a special way. After that, these subgraphs can be converted into a pyramid shape to speed up the execution of all graph operations. In this case, the number of vertices with distributive operations can increase significantly, and therefore it may be necessary to reduce their number. It follows that when transforming graphs of the second type, it is necessary to choose a specific form to which the graph will be reduced, based on the ratio of its size and the available computing resource. Thus, the proposed algorithms for transforming information graphs of various types can be effectively used in the development of computational structures based on FPGAs.

Graphs; reconfigurable computing systems; graph transformation.

Введение. Множество задач из области цифровой обработки сигналов может быть представлено в виде информационного графа. При этом важной задачей является преобразование этих графов таким образом, чтобы повысить быстродействие вычислительных структур, построенных на их основе. Примером таких структур могут служить реконфигурируемые вычислительные системы на основе программируемых логических интегральных схем [1], которые выгодно отличаются от систем, построенных на универсальных процессорах именно тем, что имеют возможность перестраивать структуру в соответствии с графом решаемой задачи. В работе [2] освещён вопрос преобразования линейных однородных информационных графов, содержащих ассоциативные операции, однако преобразования, целью которых является ускорение вычислений в соответствующих данным графам вычислительных структурах, могут быть применены не только к графам такого вида. При этом ассоциативность является важным условием этого [3, 4].

Постановка задачи. Рассмотрим последовательный информационный граф (рис. 1). В общем случае время, которое необходимо будет затратить на полное прохождение графа (последовательное выполнение всех его операций) будет рассчитываться по формуле:

$$T_G = \left(\sum_{n=1}^{N-1} l_{gn} + \sum_{k=0}^{N-2} L_{lk}\right) \cdot \tau,$$

где l_{gn} – латентности вершины g_n, $n \in [1, N]$, L_{lk} – латентность дуги l_k, $k \in [0, N-1]$, τ – длительность одного такта.

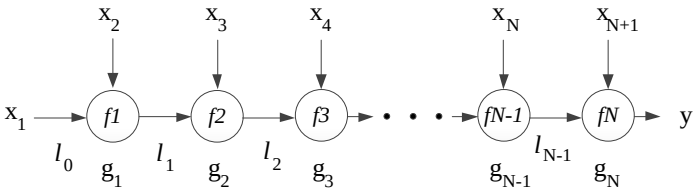


Рис. 1. Пример последовательного информационного графа

В случае если латентность всех вершин графа одинакова, а латентность соединительных дуг пренебрежимо мала, время выполнения всех операций графа будет равно

$$T_G = l_f \cdot N \cdot \tau,$$

где l_f – латентность каждой вершины графа.

Поскольку в реальных задачах значение N может достигать десятков и сотен тысяч, возникает необходимость сокращения параметра T_G. Если все входы такого графа независимы друг от друга, то мы в зависимости от прочих его свойств можем изменить структуру графа таким образом, чтобы уменьшить время выполнения всех его операций.

Предлагаемое решение. Допустим, что у нас имеется однородный последовательный информационный граф, все N вершин которого представляют собой ассоциативные операции одного типа (рис. 2). Обозначим эту операцию как *f*.

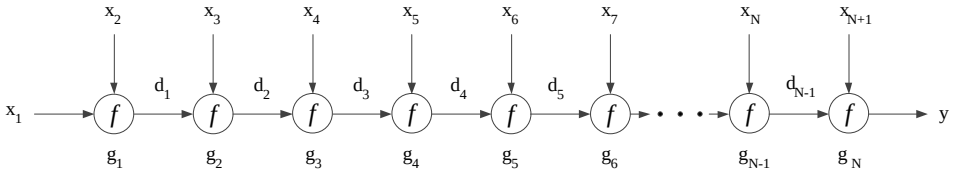


Рис. 2. Однородный последовательный информационный граф, состоящий из ассоциативных операций f

В случае если размер доступного нам оборудования составляет $R > N$, мы можем реализовать весь этот граф целиком. Для того чтобы сократить время выполнения всех операций графа, целесообразно привести его к пирамидальному виду.

Определим первую операцию – преобразование пары смежных вершин, представляющих собой одинаковые ассоциативные операции *f*, в последовательном информационном графе в параллельно-последовательную форму. Назовём её преобразованием пары ассоциативных вершин.

Согласно определению свойства ассоциативности:

$$\Big(\big((x_1 * x_2) * x_3\big) * \ldots\Big) * x_N = (x_1 * x_2) * \ldots * (x_{N-1} * x_N), \quad (1)$$

где * – рассматриваемая нами ассоциативная операция.

Введём параметр L – номер яруса, к которому относится вершина (изначально равен нулю для всех вершин).

Следовательно, мы можем произвести следующее преобразование элемента графа, включающего в себя две вершины (рис. 3):

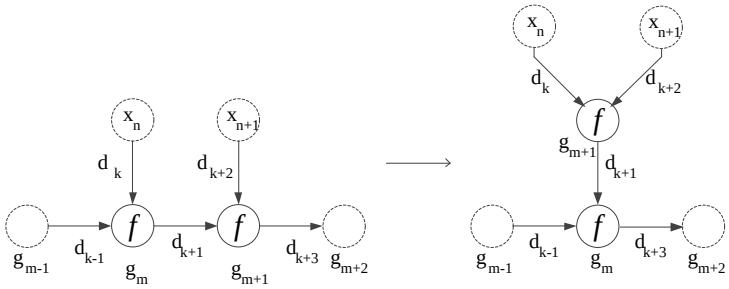


Рис. 3. Графический вид преобразования пары ассоциативных вершин

После преобразования параметр L вершины g_m будет увеличен на 1.

Для преобразования всего графа нам необходимо разбить его на пары смежных вершин и произвести над каждой парой преобразование пары ассоциативных вершин (рис. 4).

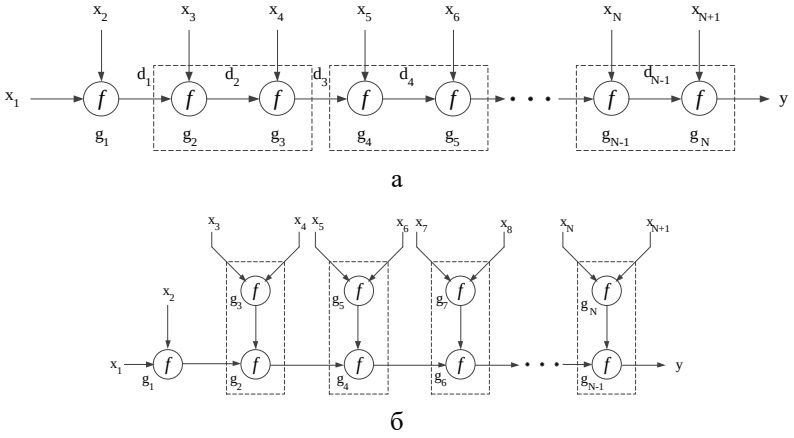


Рис. 4. Граф перед первым выполнением преобразования пары ассоциативных вершин над каждой парой (а) и после него (б)

В случае если количество вершин в графе чётное, то не преобразованными остаются первая и последняя вершины.

Затем мы повторяем преобразование для нижних вершин со значением $L = 1$, выбирая пары вершин по признаку наличия общих инцидентных дуг. Результат показан на рис. 5.

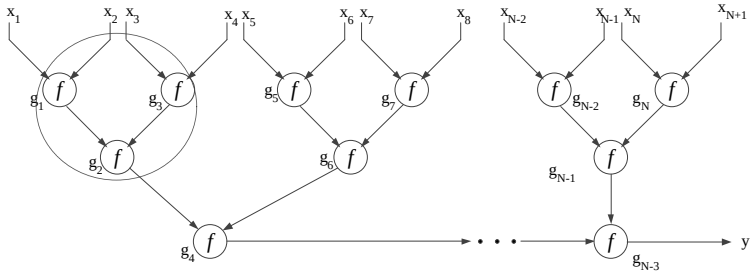


Рис. 5. Граф после второго выполнения преобразования пары ассоциативных вершин

Таким образом, алгоритм преобразования пары ассоциативных вершин для всего графа выглядит следующим образом:

1. Делим всё множество вершин на пары смежных, за исключением первой вершины и последней (в случае, если общее количество вершин в графе *N* чётное).
2. Последовательно производим преобразование пары ассоциативных вершин над парами, начиная с первой. В случае если *N* чётное, увеличиваем значение параметра *L* для последней вершины на 1.
3. Выбираем вершины с параметром *L* = 1, разбиваем их на пары по тому же принципу (не включаем в пару первую вершину и последнюю, в случае, если таких вершин чётное количество).
4. Повторяем п. 2 для этого множества.
5. П. 3 и п. 4 повторяем пока для очередного значения L количество вершин с таким значением не будет меньше трёх.

В результате граф будет приведён к пирамидальному виду (рис. 6).

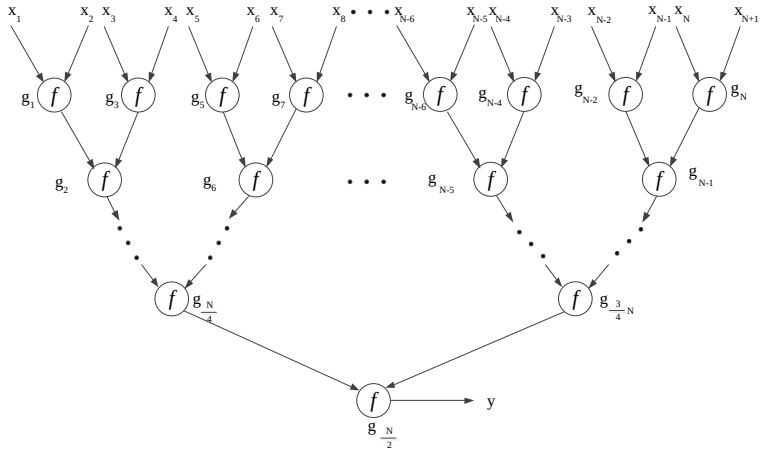


Рис. 6. Вид графа после выполнения всех доступных преобразований пар ассоциативных вершин

Время выполнения всех операций такого графа значительно сократится и составит

$$T_G = l_g \cdot \lceil log_2(N) \rceil \cdot \tau.$$

Таким образом, если количество доступного оборудования составляет $R \geq N$, мы можем ускорить полное выполнение всех операций графа в E_L раз, где

$$E_L = \frac{l_g \cdot N \cdot \tau}{l_g \cdot \lceil log_2(N) \rceil \cdot \tau} = \frac{N}{\lceil log_2(N) \rceil}.$$

В случае если $R < N$ мы не сможем реализовать весь граф целиком. В этом случае применяются методы редукции производительности, для применения которых необходимо представить граф как структуру, состоящую из подграфов (рис. 7). В этом случае множество входных вершин подграфов $x_1, x_2, \ldots, x_{N+1}$ заменяется кортежными вершинами, а дуги инцидентные связям подграфов заменяются обратными связями.

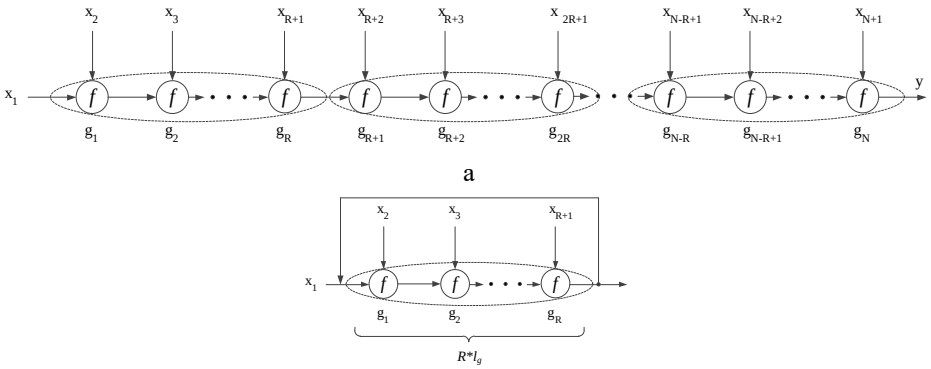


Рис. 7. Представление графа в виде совокупности подграфов (а) и вид вычислительной структуры с обратной связью (б)

В случае, если мы преобразуем таким образом исходный последовательный граф, мы столкнёмся со следующей проблемой: время заполнения вычислительного конвейера составит $R \cdot l_g$, т.к. подграфы являются информационно зависимыми и прежде, чем подавать следующий набор данных, необходимо дождаться полной обработки предыдущего, которая займёт $R \cdot l_g$ тактов. В результате мы получим скважность, равную R, что увеличит время выполнения всех операций графа в R раз.

$$T_G = R \cdot l_g \cdot N \cdot \tau.$$

Согласно теореме Иванова [5] если имеется некоторый информационный граф G, состоящий из множества взаимосвязанных изоморфных подграфов p_1, p_2, …, p_N, и ресурса системы недостаточно для аппаратной реализации всего графа G, то максимальная эффективность (удельная производительность) достигается за счет аппаратной реализации единственного подграфа p_i. В рассматриваемом случае это будет означать реализацию одной вершины (рис. 8).

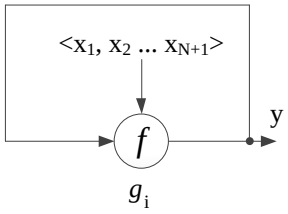


Рис. 8. Оптимальная структура графа согласно теореме Иванова

Время выполнения всех операций в этом случае составит

$$T_G = l_g \cdot (N + 1) \cdot \tau,$$

т.е. будет практически равно времени выполнения всех операций графа в случае, если $R \geq N$.

Эту величину можно уменьшить, если изоморфные подграфы, на которые разбивается исходный граф, привести к пирамидальному виду, как это было показано выше, и, что главное, обеспечить скважность, равную единице. Рассмотрим частный случай, когда параметр l_g равен единице.

Для преобразования всего графа к этой форме нам необходимо разбить его на части определённым образом. Все вершины делятся на группы, в каждую из которых *R-1* вершин. Между этими группами оставляется по одной вершине, которые составят последовательную часть графа после преобразования (рис. 9).

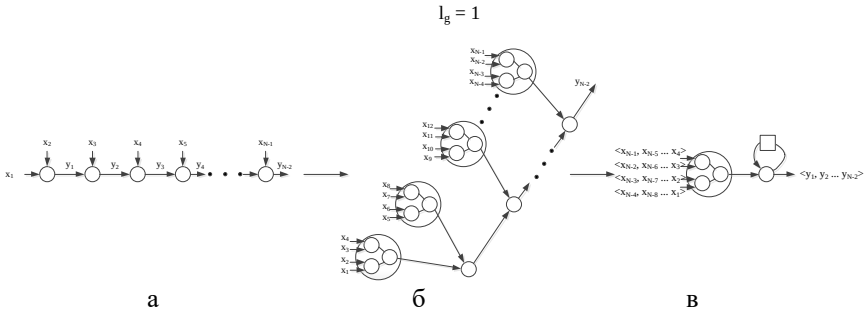


Рис. 9. Преобразование графа в параллельно-последовательную форму для случая $l_f = 1$. Исходный граф (а), преобразованный граф (б) и соответствующая ему вычислительная структура (в)

Мы выбираем первую группу вершин и преобразовываем её согласно пп. 1-5, как если бы эта группа вершин была целым графом. Затем мы переходим к следующей группе вершин и повторяем эти преобразования над ней. Одиночным вершинам присваивается значение параметра *L* равное -1.

После того, как преобразование выполнено над всеми группами, мы выделяем вторую из них и одиночную вершину, расположенную на графе слева от неё (рис. 10).

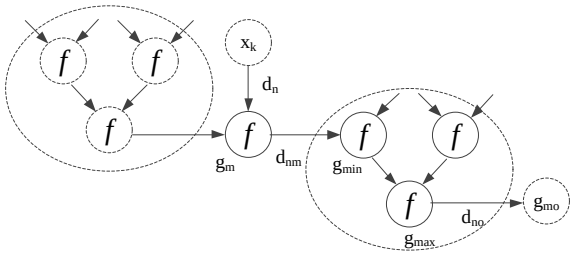


Рис. 10. Участок графа перед выполнением преобразования группы ассоциативных вершин

Результат преобразования показан на рис. 11.

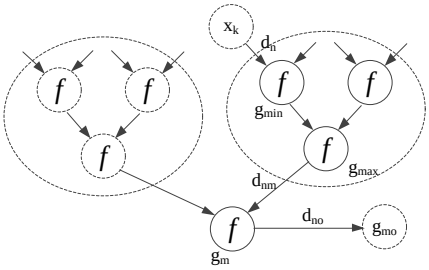


Рис. 11. Участок графа после выполнения преобразования группы ассоциативных вершин

Затем выделяем следующую пару «вершина с параметром L равным -1» и «группа вершин» и повторяем преобразование над этой парой – и так далее до обработки всех групп. Если общее количество вершин графа, который мы преобразовываем, равно

$$N = R \cdot k - 1 + n,$$

где k – количество групп, $k \in \mathbb{N}$, а $n \neq 0$, последняя группа дополняется транзитными вершинами, число которых равно $R - 1 - n$.

Таким образом, если у нас достаточно вычислительного ресурса для реализации R вычислительных узлов, выполняющих операцию f с латентностью 1, а размер графа составляет N, то граф будет разбит на $\left\lceil \frac{N}{R} \right\rceil$ базовых подграфов, каждый из которых будет содержать $R-1$ вершин в пирамидальной структуре и одну вершину, обеспечивающую обратную связь. В последней группе вершин будет содержаться $n = N - \left\lceil \frac{N}{R} \right\rceil \cdot R$ транзитных вершин, не выполняющих вычислений, но используемых для того, чтобы подграфы сохраняли регулярную структуру. Время выполнения всех операций графа будет равно

$$T_G = \left(\lceil log_2(R-1) \rceil + 1 + \left\lceil \frac{N+1}{R+1} \right\rceil \right) \cdot \tau. \quad (4)$$

Теперь рассмотрим последовательный информационный граф, состоящий из чередующихся вершин α и β, которые являются ассоциативными и операция β является дистрибутивной по отношению к операции α. Для простоты примем, что первая вершина графа является вершиной α, что количество вершин α равно количеству вершин β и что вершины разных типов чередуются (рис. 12).

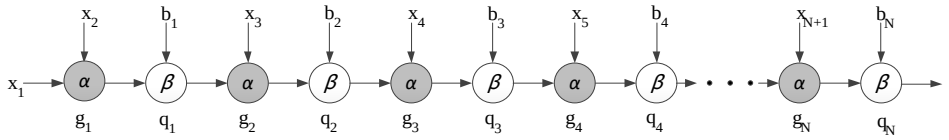


Рис. 12. Граф с чередованием двух ассоциативных операций, одна из которых является дистрибутивной по отношению к другой

Время выполнения всех операций графа будет равно

$$T_G = (l_{g\alpha} + l_{g\beta}) \cdot N \cdot \tau.$$

Для уменьшения T_G мы можем преобразовать граф в пирамидальную форму, как это было показано выше. Однако данный граф неоднороден и операции α и β в общем случае не являются ассоциативными по отношению друг к другу (хотя и являются ассоциативными по отдельности). Следовательно, прежде чем преобразовывать граф в пирамидальную форму, нам будет необходимо изменить его таким образом, чтобы выделить в нём однородные фрагменты, над которыми затем будет производиться преобразование пары ассоциативных вершин.

Согласно правилу дистрибутивности

$$(x_1 * x_2) \cdot b_1 = x_1 \cdot b_1 * x_2 \cdot b_1,$$

где $*$ – ассоциативная операция, $\cdot$ – операция, дистрибутивная по отношению к $*$.

Рассмотрим участок графа, на котором последовательно соединены две вершины: первая с операцией α, а вторая с операцией β. Мы можем преобразовать фрагмент графа, включающий в себя последовательно соединённые вершины α и β, продублировав вершину β и перенеся её с выхода вершины α, на каждый из её входов (рис. 13):

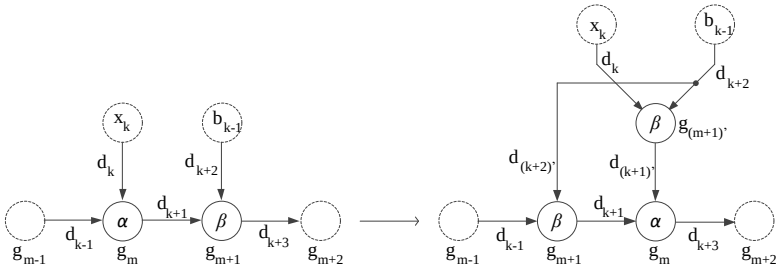


Рис. 13. Графический вид преобразования участка из двух вершин, одна из которых является дистрибутивной по отношению к другой

Последовательное применение преобразования смешанной пары вершин, когда мы перемещаем вершины β налево от вершин α и при этом с каждым переносом добавляем их копии, приведёт к тому, что исходный граф разделится на ветви, одна из которых состоит только из вершин α, а все остальные – только из вершин β (рис. 14).

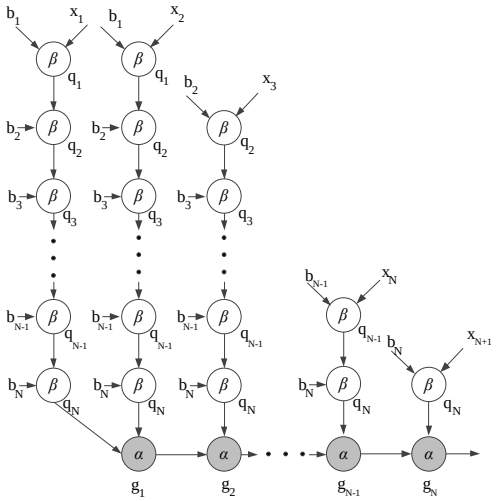


Рис. 14. Результат последовательного выполнения преобразования смешанной пары вершин над графом

Теперь, когда граф разбит на однородные фрагменты, состоящие из ассоциативных операций, мы можем выполнить над каждым из фрагментов преобразование пары ассоциативных вершин. В результате мы получим граф, состоящий из пирамиды из вершин α, на входы которой подаются выходы пирамид, состоящих из вершин β. При этом размер пирамид из вершин β будет меняться от N вершин (для двух крайне левых), до одной вершины (крайне правая пирамида) (рис. 15).

После выполнения этого преобразования и последующего выполнения преобразования над ассоциативными вершинами время выполнения всех операций графа составит

$$T_G = (l_{g\alpha} + l_{g\beta}) \cdot \lceil \log_2 N \rceil \cdot \tau.$$

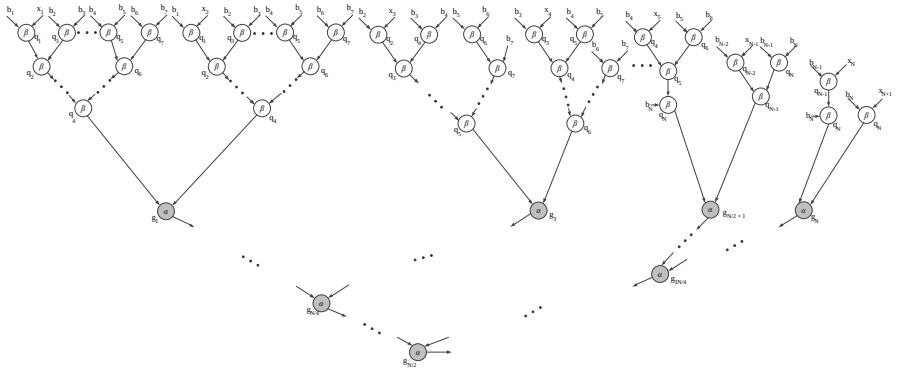
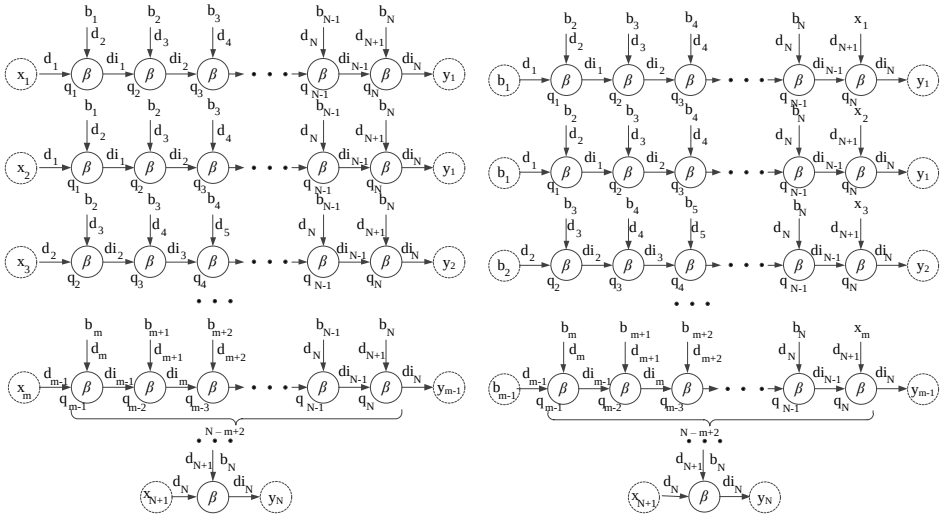


Рис. 15. Пирамидальная форма неоднородного графа

Это значение будет применимо для случая, когда у нас хватает оборудования R_α и R_β для реализации всего графа. Однако несложно заметить, что при использовании преобразования пары с дистрибутивной вершиной количество вершин N_β возрастает и в итоге становится равно $\frac{N^2+N}{2}$. N_α при этом остаётся равно N. Это приведёт к значительному повышению затрат оборудования: так, если в исходном графе было 1000 умножителей, после преобразования их количество составит 500500. В связи с этим желательно сократить количество затрачиваемого оборудования настолько, насколько это возможно.

Для сокращения затрат оборудования необходимо будет отказаться от преобразования ветвей с вершинами β к пирамидальному виду. В этом случае ветви графа с вершинами β будут иметь линейную структуру (рис. 16):



а б

Рис. 16. Структура ветвей вершин β сразу после выполнения третьего преобразования (а) и после выполнения преобразования множества вершин β (б)

Анализируя эту структуру, мы можем заметить следующее: 1. переменная x (в исходном графе входные данные недистрибутивных вершин) для каждой ветви своя; 2. две первые ветви включают в себя все переменные b (в исходном графе

входные данные дистрибутивных вершин), а каждая последующая включает на одну меньше, так, что *m*-я ветвь включает в себя переменные *b* с индексами (*m-1 … N*). Из этого можно сделать следующий вывод: для сокращения количества используемых вычислительных блоков β необходимо вынести блоки, на вход которых подаются переменные *x*, вплотную к части графа, состоящей из элементов α.

После выполнения этого преобразования надо всеми ветвями дистрибутивных вершин граф примет вид, показанный на рис. 17. Повторяющиеся вершины заштрихованы и могут быть удалены из графа.

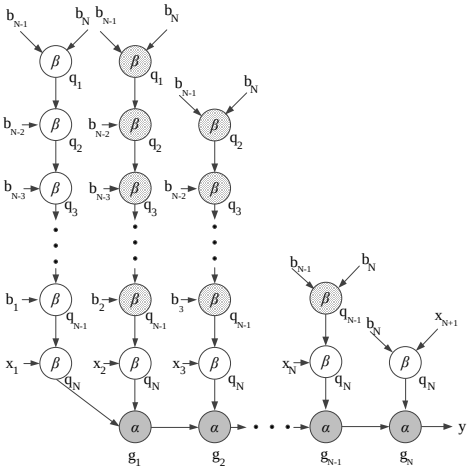


Рис. 17. Структура ветвей дистрибутивных вершин графа после перестановки вершин со входами x

После этого количество уникальных вершин β в преобразованном графе будет примерно равно *2*N-1*, а не $\frac{N^2+N}{2}$. Количество вершин α не изменится и составит *N*.

Подграф, состоящий из вершин α, может быть приведён к пирамидальной форме (рис 18).

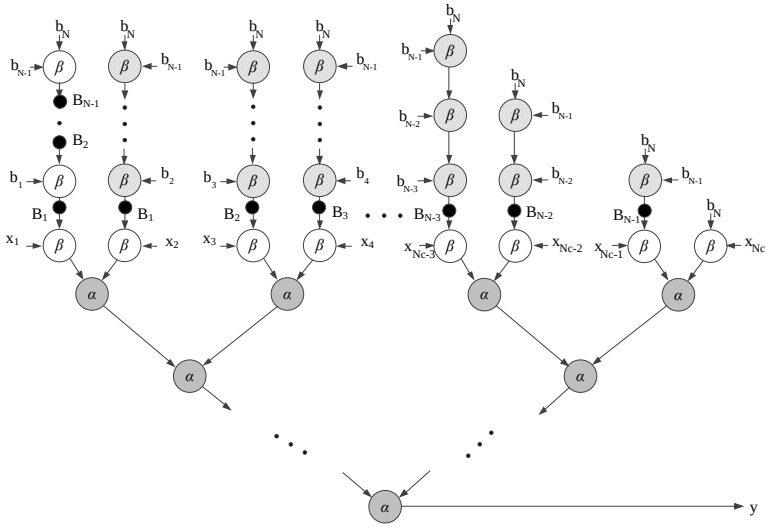


Рис. 18. Вариант линейно-пирамидальной структуры графа

В этом случае время выполнения всех операций вычислительной структуры, соответствующей этому графу, составит:

$$T_G = (l_{g\alpha} \cdot \lceil log_2 N \rceil + l_{g\beta} \cdot N) \cdot \tau.$$

В случае если $R_\alpha < N_\alpha$ и/или $R_\beta < N_\beta$, мы будем вынуждены разбить граф на подграфы и реализовывать по одному из них единовременно.

Рассмотрим множество вершин $\{g_1, g_2, \ldots g_N\}$ на рис. 14. В случае если граф будет охвачен петлёй обратной связи, желательно минимизировать длину цепочки, соединяющей вершину g_1 и выход y. Следовательно необходимо произвести следующее преобразование (рис. 19):

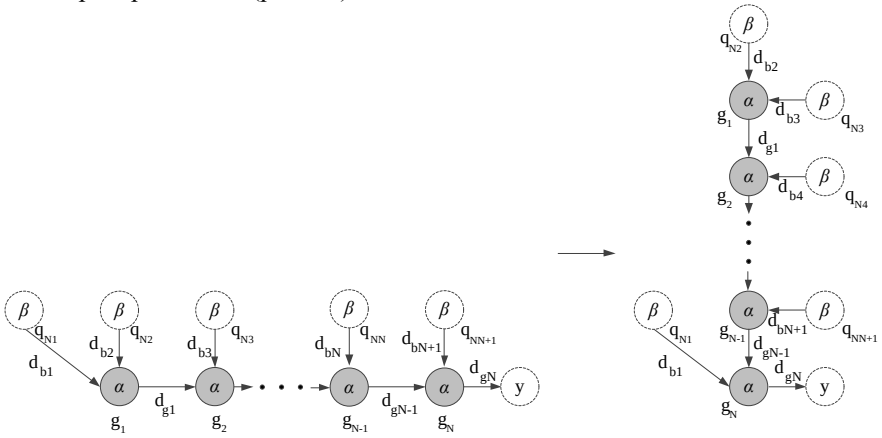


Рис. 19. Графический вид преобразования множества вершин α

Последовательное применение преобразования смешанной пары вершин и последующая перестановка вершин, с целью сократить повторяющиеся вершины, приведёт подграф к следующему виду:

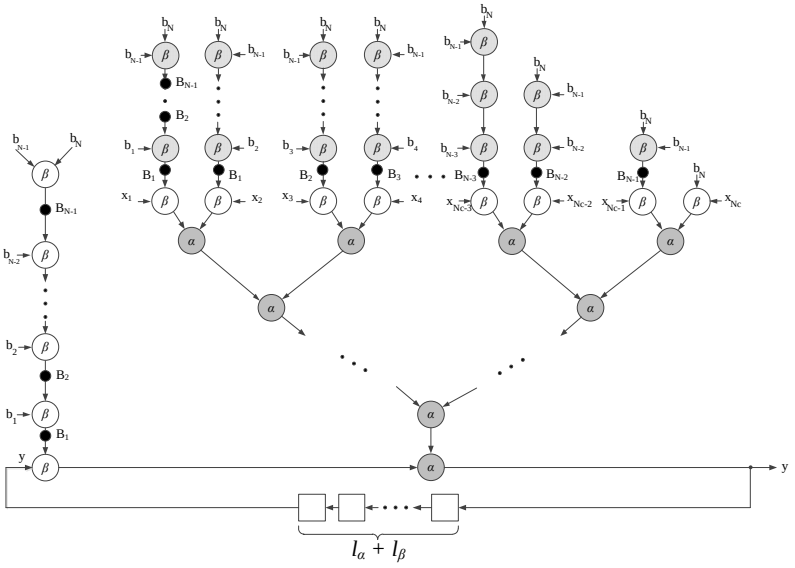


Рис. 20. Вид подграфа после преобразований

Согласно теореме Иванова максимальная производительность в данном случае будет достигаться при реализации фрагмента, состоящего из двух последовательно соединённых вычислительных блоков: α и β (рис. 21).

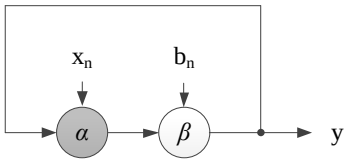


Рис. 21. Наименьший смешанный подграф, соответствующий оптимальной вычислительной структуре согласно теореме Иванова

Время реализации всех операций графа в этом случае составит:

$$T_G = (l_\alpha + l_\beta + \frac{N}{2}) \cdot \tau \cdot S,$$

где $S = l_\alpha + l_\beta$ – скважность, с которой будут подаваться данные, равная латентности подсистемы, реализующей наименьший смешанный подграф: последовательно соединённые блоки α и β;

$N = N_\alpha + N_\beta$ – общее число вершин в исходном графе.

Общая латентность подсистемы, равная $l_\alpha + l_\beta$, определит также время заполнения вычислительного конвейера, которое, в свою очередь, определит минимальную скважность, с которой будут подаваться входные данные. Подача данных плотным потоком здесь будет невозможна, поскольку предыдущая партия должна будет закончить обработку и попасть на вход блока α до того, как на другой его вход будет подана следующая партия.

Очевидно, что максимальным членом этого выражения будет являться $\frac{N}{2}$ и что значение этого выражения никак не зависит от доступного нам вычислительного ресурса. Это объясняется тем, что если мы реализуем большее количество пар блоков α и β, то скважность, с которой придётся подавать входные данные, возрастёт пропорционально. Более того, при реализации такой вычислительной структуры возникнут дополнительные задержки. Таким образом кажется, что мы никак не можем ускорить выполнение всех операций такого графа, в случае, если у нас не хватает оборудования на то, чтобы реализовать его целиком.

Однако лучший результат может быть достигнут в случае, если мы реализуем фрагмент вида, представленного на рис. 30, состоящий из M_α вершин α и M_β вершин β, где $M_\alpha = l_\alpha + l_\beta$, $M_\beta = 2 \cdot (l_\alpha + l_\beta)$. В этом случае раз в S тактов будут подаваться $2 \cdot (l_\alpha + l_\beta)$ данных, что будет означать ускорение подачи данных в $l_\alpha + l_\beta$ раз по сравнению с предыдущим вариантом. В результате время выполнения всех операций станет равно:

$$T_G = (L + \left\lceil \frac{N}{2 \cdot (l_\alpha + l_\beta)} \right\rceil) \cdot \tau \cdot S,$$

где $L = (l_\alpha + l_\beta) \cdot l_\beta + \lceil log_2(l_\alpha + l_\beta) \rceil \cdot l_\alpha$ – латентность критического (т.е. наибольшего) пути внутри вычислительной структуры.

Заключение. Таким образом, в зависимости от соотношения числа доступных нам вычислительных блоков обоих типов, их латентности и общего числа вершин в рассматриваемом графе, мы можем выделить несколько вариантов построения вычислительной структуры, каждый из которых выполняет все операции графа со своей скоростью. Это позволит выбирать оптимальный способ преобразования графа в зависимости от соотношения числа его вершин и доступных нам вычислительных ресурсов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Левин И.И., Дордопуло А.И.* К вопросу об автоматическом создании параллельных прикладных программ для реконфигурируемых вычислительных систем // Вычислительные технологии. – 2020. – Т. 25, № 1. – С. 66-81.
2. *Левин И.И., Дордопуло А.И., Писаренко И.В., Михайлов Д.В.* Представление графов с ассоциативными операциями на языке программирования SET@L // Известия ЮФУ. Технические науки. – 2020. – № 3. – С. 98-111.
3. *Кнут Д.Э.* Искусство программирования. Т.4, А. Комбинаторные алгоритмы. Ч. 1: пер. с англ. – М.: ООО «И. Д. Вильямс», 2013. – 960 с.
4. *Новиков Ф.* Дискретная математика. – 3-е изд. – СПб.: Питер, 2019. – 496 с.
5. *Иванов А.И.* Методы и средства создания эффективного параллельно-конвейерного программного обеспечения вычислительных систем, построенных на основе ПЛИС-технологии: дисс. … канд. техн. наук, по специальности: 05.13.11 “Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей”, научный руководитель: чл.-корр. РАН, д.т.н., проф. Каляев И.А., дис. совет ТРТУ Д 212.259.05, 2005. – 182 с.
6. Задача суммирования элементов массива. – Лаборатории Параллельных информационных технологий НИВЦ МГУ. – URL: https://parallel.ru/fpga/Summ2 (дата обращения: 27.04.2020).
7. *Ефимов С.С.* Обзор методов распараллеливания алгоритмов решения некоторых задач вычислительной дискретной математики // Математические структуры и моделирование. – 2007. – Вып. 17. – С. 72-93.
8. *Tessier R., Pocek K., DeHon A.* Reconfigurable Computing Architectures // Proceedings of the IEEE. – 2015. – Vol. 103, No. 3. – P. 332-354.
9. *Mittal S., Vetter J.* A survey of CPU-GPU heterogeneous computing techniques // ACM Computing Surveys. – 2015. – Vol. 47. – Art. 69.
10. *Waidyasooriya H.M., Hariyama M., Uchiyama K.* Design of FPGA-Based Computing Systems with OpenCL. – Cham: Springer, 2018. – 126 p.
11. *Reinhard Diestel.* Graph Theory. Springer-Verlag, Heidelberg // Graduate Texts in Mathematics. – 2016. – Vol. 173. – 447 p. – ISBN 978-3-662-53621-6.
12. *Edward A. Bender.* Lists, Decisions and Graphs. With an Introduction to Probability. S. Gill Williamson. – 2010. – 251 p.
13. *Trudeau, Richard J.* Introduction to graph theory. – Dover Publications, Inc. New York, 1993. – 224 p.
14. *Старченко А.В., Берцун В.Н.* Методы параллельных вычислений. – Томск: Изд-во Том. ун-та, 2013. – 223 с.
15. *Левин И.И., Дордопуло А.И.* К вопросу об автоматическом создании параллельных прикладных программ для реконфигурируемых вычислительных систем // Вычислительные технологии. – 2020. – Т. 25, № 1. – С. 66-81.
16. *Каляев А.В., Левин И.И.* Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.
17. *Левин И.И., Дордопуло А.И., Гудков В.А. и др.* Средства программирования реконфигурируемых и гибридных вычислительных систем на основе ПЛИС // XIII Междунар. конф. «Параллельные вычислительные технологии» (ПаВТ-2019): Короткие статьи и описания плакатов. – Челябинск: Изд. центр ЮУрГУ, 2019. – С. 299-312.
18. *Дасгупта С., Пападимитриу Х., Вазирани У.* Алгоритмы: пер. с англ. / под ред. А. Шеня. – М.: МЦНМО, 2014. – 320 с.
19. *Харари Ф.* Теория графов. – М.: Мир, 1973. – 300 с.
20. *Кормен Т.М. и др.* Часть VI. Алгоритмы для работы с графами. Алгоритмы: построение и анализ = Introduction to Algorithms. – 2-е изд. – М.: Вильямс, 2006. – 1296 с.

REFERENCES

1. *Levin I.I., Dordopulo A.I.* K voprosu ob avtomaticheskom sozdanii parallel'nykh prikladnykh programm dlya rekonfiguriruemykh vychislitel'nykh sistem [On the problem of automatic development of parallel applications for reconfigurable computer systems], *Vychislitel'nye tekhnologii* [Computational Technologies], 2020, Vol. 25, No. 1, pp. 66-81.

2. *Levin I.I., Dordopulo A.I., Pisarenko I.V., Mikhaylov D.V.* Predstavlenie grafov s assotsiativnymi operatsiyami na yazyke programmirovaniya SET@L [Description of graphs with associative operations in SET@L programming language], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2020, No. 3, pp. 98-111.
3. Knut D.E. Iskusstvo programmirovaniya. T.4, A. Kombinatornye algoritmy [The Art of Computer Programming. Vol. 4, A. Combinatorial Algorithms]. Part 1: transl. from engl. Moscow: OOO «I. D. Vil'yams», 2013, 960 p.
4. *Novikov F.* Diskretnaya matematika [Discrete Mathematics]. 3rd ed. Saint Petersburg: Piter, 2019, 496 p.
5. *Ivanov A.I.* Metody i sredstva sozdaniya effektivnogo parallel'no-konveyernogo programmnogo obespecheniya vychislitel'nykh sistem, postroennykh na osnove PLIS-tekhnologii: diss. … kand. tekhn. nauk, po spetsial'nosti: 05.13.11 "Matematicheskoe i programmnoe obespechenie vychislitel'nykh mashin, kompleksov i komp'yuternykh setey" [Methods and tools for creating effective parallel pipeline software for computing systems based on FPGA technology: diss. … cand. of eng. sc., specialty: 05.13.11 "Mathematical and software support for computers, complexes and computer networks"], scientific director: Corresponding Member of RAS, d.t.s., proff. Kalyaev I.A., dis. council TSREU D 212.259.05, 2005, 182 p.
6. Zadacha summirovaniya elementov massiva [Problem of Array Elements' Summation], The Laboratory of Parallel Information Technologies of the Research Computing Center of the Moscow State University. Available at: https://parallel.ru/fpga/Summ2 (accessed 27 April 2020).
7. *Efimov S.S.* Obzor metodov rasparallelivaniya algoritmov resheniya nekotorykh zadach vychislitel'noy diskretnoy matematiki [Review of Parallelizing Methods for Algorithms Aimed at Solution of Certain Problems of Computational Discrete Mathematics], *Matematicheskie struktury i modelirovanie* [Mathematical Structures and Modeling], 2007, Issue 17, pp. 72-93.
8. *Tessier R., Pocek K., DeHon A.* Reconfigurable Computing Architectures, *Proceedings of the IEEE*, 2015, Vol. 103, No. 3, pp. 332-354.
9. *Mittal S., Vetter J*. A survey of CPU-GPU heterogeneous computing techniques, *ACM Computing Surveys*, 2015, Vol. 47, Art. 69.
10. *Waidyasooriya H.M., Hariyama M., Uchiyama K.* Design of FPGA-Based Computing Systems with OpenCL. Cham: Springer, 2018, 126 p.
11. *Reinhard Diestel.* Graph Theory. Springer-Verlag, Heidelberg, *Graduate Texts in Mathematics*, 2016, Vol. 173, 447 p. ISBN 978-3-662-53621-6.
12. *Edward A. Bender.* Lists, Decisions and Graphs. With an Introduction to Probability. S. Gill Williamson, 2010, 251 p.
13. *Trudeau, Richard J.* Introduction to graph theory. Dover Publications, Inc. New York, 1993, 224 p.
14. *Starchenko A.V., Bertsun V.N.* Metody parallel'nykh vychisleniy [Methods of Parallel Computing]. Tomsk: Izd-vo Tom. un-ta, 2013, 223 p.
15. *Levin I.I., Dordopulo A.I.* K voprosu ob avtomaticheskom sozdanii parallel'nykh prikladnykh programm dlya rekonfiguriruemykh vychislitel'nykh sistem [On the problem of automatic development of parallel applications for reconfigurable computer systems], *Vychislitel'nye tekhnologii* [Computational Technologies], 2020, Vol. 25, No. 1, pp. 66-81.
16. *Kalyaev A.V., Levin I.I.* Modul'no-narashchivaemye mnogoprotsessornye sistemy so strukturno-protsedurnoy organizatsiey vychisleniy [Modular-Expandable Multiprocessor Systems with Structural and Procedural Organization of Calculations]. Moscow: Yanus-K, 2003, 380 p.
17. *Levin I.I., Dordopulo A.I., Gudkov V.A. i dr.* Sredstva programmirovaniya rekonfiguriruemykh i gibridnykh vychislitel'nykh sistem na osnove PLIS [Tools for Programming of Reconfigurable and Hybrid Computer Systems Based on FPGAs], *XIII Mezhdunar. konf. «Parallel'nye vychislitel'nye tekhnologii» (PaVT-2019): Korotkie stat'i i opisaniya plakatov* [XIII International Conference «Parallel computational technologies» (PCT'2019), short papers and poster descriptions]. Chelyabinsk: Izd. tsentr YuUrGU, 2019, pp. 299-312.
18. *Dasgupta S., Papadimitriu Kh., Vazirani U.* Algoritmy [Algorithms]: transl. from engl., ed. by A. Shenya. Moscow: MTSNMO, 2014, 320 p.
19. *Kharari F.* Teoriya grafov [Graph Theory]. Moscow: Mir, 1973, 300 p.
20. *Kormen T.M. i dr. Chast' VI.* Algoritmy dlya raboty s grafami. Algoritmy: postroenie i analiz = Introduction to Algorithms [Part IV. Algorithms for working with graphs, Algorithms: construction and analysis = Introduction to Algorithms]. 2nd ed. Moscow: Vil'yams, 2006, 1296 p.

Статью рекомендовал к опубликованию д.т.н. Э.В. Мельник.

Михайлов Денис Васильевич – ООО "Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров";e-mail: mixailow.den@gmail.com; 347900, г. Таганрог, ул. Фрунзе, 61, кв. 11; тел.: 89287502869; аспирант.

Mikhailov Denis Vasilevich – LLC "Scientific-research center of supercomputers and Neurocomputers"; e-mail: mixailow.den@gmail.com; 61, Frunze street, ap. 11, Taganrog, 347900, Russia; phone: +79287502869; postgraduate student.