Ковалев Владислав Владимирович – Южный федеральный университет; e-mail: vlad.kovalev94@mail.ru; г. Таганрог, Россия; тел.: +79525864492; кафедра вычислительной техники; аспирант.

Сергеев Николай Евгеньевич – e-mail: nesergeev@sfedu.ru; тел.: +79281742585; кафедра вычислительной техники; д.т.н.; профессор.

Kovalev Vladislav Vladimirovich – Southern Federal University**;** e-mail: vlad.kovalev94@mail.ru; Taganrog, Russia; phone: +79525864492; the department of computer science; post-graduate student.

Sergeev Nikolay Evgenievich – e-mail: nesergeev@sfedu.ru; phone: +79281742585; the department of computer science; dr. of eng. sc.; professor.

УДК 004.472.43 DOI 10.18522/2311-3103-2021-5-154-168

С.А. Дудко, И.И. Левин

МЕТОДЫ ПРЕОБРАЗОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР С ОБРАТНЫМИ СВЯЗЯМИ ДЛЯ ЭФФЕКТИВНОЙ РЕАЛИЗАЦИИ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

В настоящее время для решения задач на реконфигурируемых вычислительных системах используются различные системы автоматизированного проектирования. В большинстве случаев они состоят из двух основных компонент: компилятора (транслятора), переводящего текст исходной программы в графовую информационно-вычислительную структуру, и синтезатора, размещающего ее в архитектуре программируемых логических интегральных схем. Существующие синтезаторы, как правило, обрабатывают информационно-вычислительную структуру без комплексной оптимизации. Поэтому полученное решение прикладной задачи может содержать неэффективные фрагменты, снижающие быстродействие прикладной программы. Наиболее распространёнными примерами неэффективных вычислительных структур являются фрагменты, реализующие рекурсивные выражения, так как они снижают быстродействие прикладной программы. В статье предложены методы преобразования рекурсивных выражений (фрагментов с обратными связями), которые позволяют в автоматическом режиме сократить интервал обработки данных при решении прикладных задач на реконфигурируемых вычислительных системах. В основе методов лежат информационно-эквивалентные преобразования информационно-вычислительной структуры исходной задачи. Для каждого преобразования определен набор правил, которым должны удовлетворять операционные вершины вычислительной структуры. Применение правил позволяет выполнять эквивалентные преобразования не только над простыми структурами данных, такими как числа, но также и над более сложными структурами (матрицами, векторами, тензорами и т.п.). По результатам моделирования разработанные методы преобразования информационно-вычислительных структур с обратными связями позволяют сократить время решения прикладных задач примерно в 2–5 раз, за счет сокращения интервала обработки данных. Предложенные методы реализованы в прототипе оптимизирующего синтезатора информационно-вычислительных структур.

Информационно-эквивалентные преобразования; оптимизирующий синтезатор; реконфигурируемые вычислительные системы.

S.A. Dudko, I.I. Levin

TRANSFORMATION METHODS OF COMPUTING STRUCTURE WITH FEEDBACKS FOR EFFECTIVE IMPLEMENTATION ON RECONFIGURABLE COMPUTING SYSTEMS

At present, various computer-aided (CAD) systems are used for solving tasks on reconfigurable computing systems (RCS). In most cases, they consist of two main parts: a compiler (translator), which translates the source code of a program into a graph-like information and computing structure, and a synthesizer, which maps it on an FPGA architecture. As a rule, existing synthesiz-

ers process computing structures without any complex optimization. Therefore, the solution, generated by the synthesizer, may contain inefficient fragments, which decrease a task solution speed. The most common examples of inefficient computing structures are fragments which implement recursive expressions. The paper proposes transformation methods for recursive expressions (fragments with feedbacks), which allow automatically reduce the data supply interval when solving tasks on reconfigurable computing systems. The methods are based on information-equivalent transformations of the computing structure of the original task. For each transformation defined a set of rules that must be satisfied by the vertices of the computing structure. Applying rules allows to perform equivalent transformations not only on simple data structures such as numbers, but also on more complex structures (matrices, vectors, tensors, etc.). On the base of the simulation results, the developed transformation methods of computing structures with feedbacks allow to reduce the task solving time about 2–5 times by reducing the data supply interval. The proposed methods are implemented in a prototype of optimizing synthesizer.

Information-equivalent transformations; optimizing synthesizer; reconfigurable computing systems.

Введение. Для решения вычислительно трудоёмких задач за приемлемое время требуются высокопроизводительные вычислительные системы (ВС). Реконфигурируемые вычислительные системы (РВС) [1, 2], построенные на базе программируемых логических интегральных схем (ПЛИС), доказали свою высокую эффективность при решении задач, требующих обработки больших объемов информации [3-4]. Поскольку создание прикладных программ для ПЛИС является длительной и трудоемкой задачей, то при разработке схемотехнический решений на их основе, как правило, применяются различные системы автоматизированного проектирования (САПР). Основными частями данных САПР, чаще всего, являются компилятор (транслятор), выполняющий построение вычислительной структуры на основе исходного текста программы [1], и синтезатор, занимающийся компоновкой и размещением вычислительной структуры на кристаллах ПЛИС [5].

Существующие синтезаторы САПР [6–8], как правило, используют ряд эвристик и оптимизаций, для сокращения аппаратных затрат, отдельных фрагментов вычислительной структуры, но не применяют комплексную оптимизацию. Поэтому полученное синтезатором решение в ряде случаев может содержать неэффективные фрагменты, существенно снижающие производительность прикладной программы. Наиболее часто такие фрагменты характеризуются увеличенным (по сравнению с остальными подзадачами) интервалом обработки данных (скважностью) вычислительной структуры [9]. Как правило, такие случаи возникают при реализации рекурсивных выражений и выражений с косвенной адресацией, вычислительная структура которых содержит обратные связи.

Для сокращения времени решения задачи эти структуры необходимо оптимизировать. Оптимизация вычислительных структур – это задача высококвалифицированного инженера-схемотехника, требующая значительного времени на анализ, реализацию и тестирование.

В связи с этим возникает необходимость в разработке новых методов и средств для структурных преобразований неэффективных конструкций в автоматическом режиме, что существенно сократит время решения задачи и время создания программ для РВС, упростит процесс разработки и повысит удельную производительность прикладной задачи.

Информационно-эквивалентные преобразования. Вычислительная (информационно-вычислительная) структура задачи может быть представлена в виде графа G (V, D), который содержит в себе множество вершин V и множество дуг D. Множество вершин V в свою очередь может быть представлено как тройка V (Q, X, Y), где Q – множество операционных вершин, X – множество входных информационных вершин, Y – множество выходных информационных вершин. Операционная вершина вычислительной структуры представляет собой библио-

течный элемент, выполняющий заданную операцию над данными. Информационная вершина описывает данные, над которыми производятся вычисления в операционных вершинах. Дуги информационного графа определяют информационной обмен между вершинами графа.

Под информационно-эквивалентным преобразованием будем понимать такое преобразование, которое позволяет изменять вычислительную структуру решаемой задачи с выполнением следующего условия: при одинаковом входном воздействии (одинаковом наборе входных данных) на выходах обоих вычислительных структур (исходной и модернизированной) будет получен одинаковый (эквивалентный) результат. Далее будем называть информационно-эквивалентные преобразования просто эквивалентными.

Информационно-эквивалентные преобразования должны удовлетворять принципу биективности [10, 11], т.е. быть взаимно однозначными.

Для каждого эквивалентного преобразования должен быть определен набор правил, которым должны удовлетворять вершины информационно-вычислительной структуры, над которыми производятся преобразования. Эквивалентные преобразования могут образовывать цепочки преобразований (последовательность различных эквивалентных преобразований), которые в свою очередь могут объединяться в сценарий (алгоритм преобразования более сложных фрагментов информационного графа).

Для применения информационно-эквивалентных преобразований вычислительную структуру задачи целесообразно представить в виде плоского (неиерархичного) графа, что позволит находить последовательно соединенные вершины графа. Поиск последовательностей вершин в графе является более простой операцией в отличии от поиска подобных фрагментов в тексте исходной программы, где зависимости между переменными и операциями могут быть распределены по различным частям текста. Поэтому поиск зависимостей и преобразование текста являются трудоемкой задачей, так как компилятору (транслятору) необходимо следить за множеством факторов (корректностью индексов операндов, раскрытием скобок, подстановкой результатов одного выражения в другое и т.д.). В связи с этим, проведение преобразований над информационно-вычислительной структурой, представленной в виде плоского графа, на этапе синтеза является предпочтительным решением.

Базовые информационно-эквивалентные преобразование. Современные ПЛИС имеют большой вычислительный ресурс, который продолжает расти, но при этом количество доступных пользователю каналов информационного обмена практически не изменилось за последние 15 лет. Поэтому при решении задач на ПЛИС необходимо эффективно использовать каналы информационного обмена. Для этого разработаны различные методы, позволяющие загружать разнородные данные по одному каналу, а затем разделять их внутри вычислительной структуры. В этом случае возникают ситуации, когда данные поступают на вершины вычислительной структуры не плотным потоком (каждый такт), а с некоторым интервалом подачи данных, большим единицы. Величина интервала подачи данных зависит от реализации вычислительной структуры и кратно увеличивает время решения задачи.

Высокий интервал обработки данных (скважность, S) является одной из основных причин, снижающих производительность прикладной программы. Интервал обработки данных характеризует плотность формирования данных во входном/выходном потоке. При S=1 поток является плотным (новые данные формируются каждый такт); S>1 свидетельствует о наличии разрывов (задержек) при формировании данных.

Наиболее часто встречающимися структурами, увеличивающими интервал обработки данных, являются структуры с обратными связями. Увеличение интервала обработки данных подобных структур обусловлено необходимостью дожи-

даться результата по обратной связи для обработки текущего данного. Это приводит к замедлению не только фрагмента с обратной связью, но и всей вычислительной структуры, так как поток данных в последующих фрагментах тоже будет зависеть от интервала формирования выходных данных в обратной связи. Пример вычислительной структуры с обратной связью показан на рис. 1.

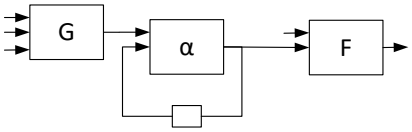


Рис. 1. Фрагмент вычислительной структуры с обратной связью

Каждая операционная вершина вычислительной структуры характеризуется собственным интервалом обработки данных (количество тактов, необходимое для выполнения операции). Интервал обработки данных в обратной связи может быть рассчитан как отношение длины критического пути (максимальный суммарный интервал обработки данных операционными вершинами, образующими обратную связь) к количеству регистров в обратной связи.

Рассмотрим набор базовых информационно-эквивалентных преобразований, основанных на математических законах ассоциативности и дистрибутивности [12, 13], которые будут использоваться для преобразования вычислительных структур с обратными связями.

1) Эквивалентное преобразование ассоциативных операционных вершин. Применение преобразования, схема которого представлена на рис. 2, позволяет выносить одну из операционных вершин за пределы обратной связи, уменьшая длину критического пути и, тем самым, снижая интервал обработки данных. Данное преобразование выполняется, если операционная вершина φ, является бинарной (имет два входа и один выход) и удовлетворяет условию ассоциативности [13]:

$$(a\ \varphi\ b)\varphi\ c = a\ \varphi\ (b\ \varphi\ c).$$

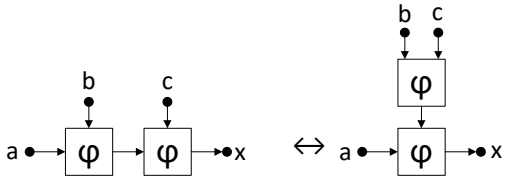


Рис. 2. Информационно-эквивалентное преобразование ассоциативных операционных вершин

2) Эквивалентное преобразование дистрибутивных операционных вершин. Использование данного преобразования позволяет изменять порядок следования вершин в вычислительной структуре задачи. Изменение порядка следования операционных вершин, показанное на рис. 3, необходимо проводить тогда, когда две ассоциативные вершины отделены друг от друга дистрибутивной, затем может быть применено преобразование ассоциативных вершин. Преобразование дистрибутивных вершин является эквивалентным при условии, что рассматриваемые вершины β и φ являются бинарными, а также дистрибутивными относительно множества K, над которым выполняются вычисления [13]:

$$(a\ \beta\ b)\ \varphi\ (a\ \beta\ c) = (b\ \varphi\ c)\ \beta\ a.$$

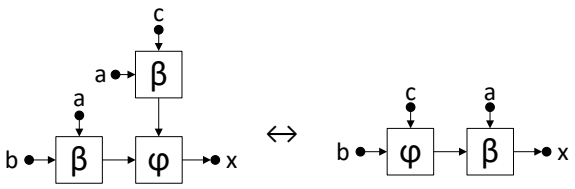


Рис. 3. Информационно-эквивалентное преобразование дистрибутивных операционных вершин

3) Пространственная свертка/развертка последовательности ассоциативных вершин. С помощью данного преобразования возможно представить последовательность ассоциативных вершин как одну общую вершину с обратной связью (рис. 4). Это позволяет разворачивать обратные связи в пространстве, и затем проводить над ними оптимизирующие преобразования, либо сворачивать набор ассоциативных вершин для экономии аппаратного ресурса.

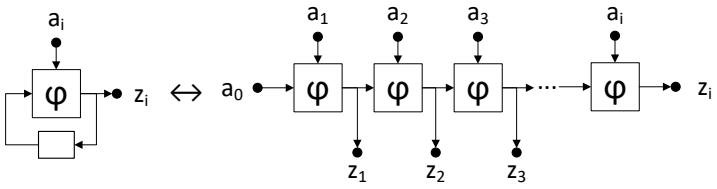


Рис. 4. Информационно-эквивалентное преобразование «свертки/развертки» последовательности ассоциативных вершин

На основе рассмотренных выше базовых информационно-эквивалентных преобразований можно предложить методы модернизации вычислительных структур с обратными связями, соответствующие рекурсивным выражениям.

Применение метода автоподстановки для преобразования рекурсивных выражений в вычислительной структуре. Одним из методов преобразования рекурсивных выражений, состоящих из дистрибутивных операций, является метод автоподстановки [14, 15], применяемый в конвейерных вычислениях над текстовым представлением задачи. Особенностью метода автоподстановки является то, что он может быть применен многократно к одному и тому же выражению, до тех пор, пока не будет получен необходимый результат. Используя концепцию метода автоподстановки как преобразования над математическими формулами, произведем подобные преобразования над информационно-вычислительной структурой задачи в графовом виде.

Рассмотрим следующий фрагмент вычислительной структуры (рис. 5).

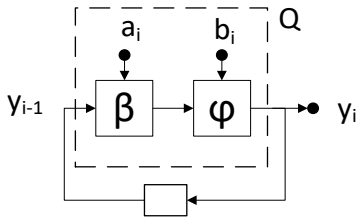


Рис. 5. Фрагмент вычислительной структуры с обратной связью

На рис. 5 операционные вершины β и φ представляют собой любые операции (либо блок операций), выполняемые над любыми структурами данных (числами, матрицами, векторами, тензорами и т.п.), для которых справедливы правила дистрибутивности и ассоциативности. В общем случае преобразование автоподстановки выполняется над множеством K, образующим полукольцо [13, 16, 17] относительно операционных вершин β и φ, то есть данные вершины удовлетворяют следующим условиям:

1. Для операционной вершины φ выполняются коммуникативный и ассоциативный законы;
2. Для операционной вершины φ определен нейтральный элемент n^0, такой что:

$$a\ \varphi\ n^0 = n^0\ \varphi\ a = a;$$

3. Операционная вершина β образует полугруппу относительно множества K, над которым она определена (выполняется правило ассоциативности);
4. Операционные вершины β и φ дистрибутивны между собой.

Обозначим выделенный пунктирной линией фрагмент вычислительной структуры, показанный на рис. 5, как Q. Далее, будем считать, что интервал обработки данных вершинами β и φ равняется n и m соответственно. В таком случае интервал поступления данных на входы операционных вершин вычислительной структуры будет равняться (n + m), что приводит к снижению быстродействия прикладной программы в (n + m) раз. Очевидно, что для повышения быстродействия, необходимо уменьшить интервал поступления данных на входы операционных вершин, образующих обратную связь. Для этого предлагается распараллелить данный фрагмент вычислительной структуры так, чтобы одновременно выполнялся расчет (n+m) значений.

Примем n = m = 1, тогда интервал поступления данных исходного фрагмента вычислительной структуры (рис. 5) будет равен 2. Преобразуем исходный фрагмент к следующему виду (рис. 6), воспользовавшись преобразованием свертки/развертки последовательности ассоциативных операционных вершин – добавив после текущего выходного сигнала (n+m-1) подграфов Q.

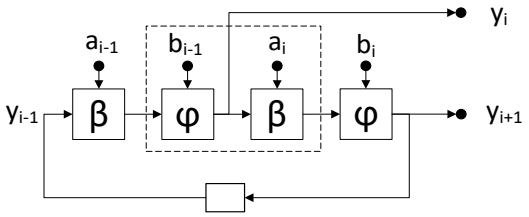


Рис. 6. Структурное расширение исходного фрагмента вычислительной структуры

Как можно заметить, для фрагмента вычислительной структуры, выделенного пунктирной линией (рис. 6), можно применить преобразование над дистрибутивными вершинами, предварительно заменив ветвящийся выход вершины φ на одиночный. Для этого перенесем выход с текущей вершины на предыдущую продублировав вычисления для вершины φ (рис. 7).

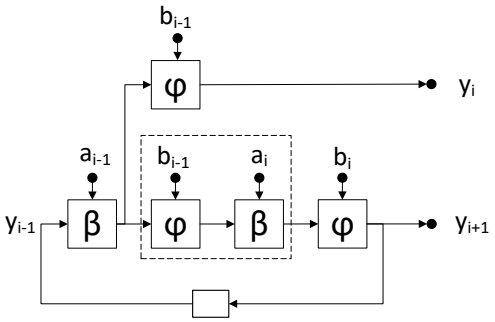


Рис. 7. Перенос ветвящегося выхода через операционную вершину

Далее, используя преобразование дистрибутивных вершин, изменим выделенный пунктирной линией фрагмент (рис. 7) вычислительной структуры:

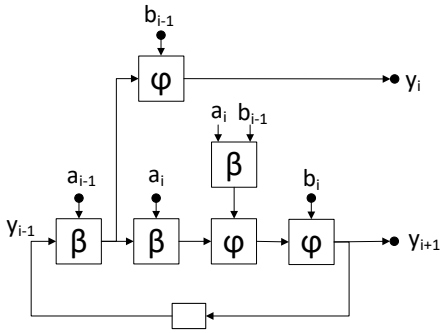


Рис. 8. Фрагмент вычислительной структуры после применения преобразования дистрибутивных операционных вершин

Упростим полученную вычислительную структуру (вынесем из обратной связи ассоциативные вершины), используя рассмотренные ранее эквивалентные преобразования:

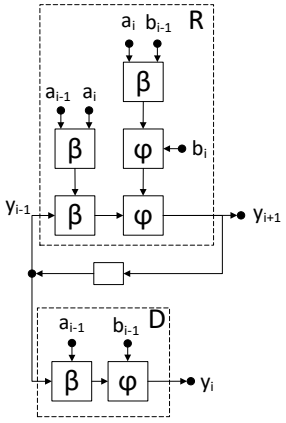


Рис. 9. Фрагмент вычислительной структуры после применения преобразования ассоциативных операционных вершин

Обозначим выделенные пунктиром фрагменты (рис. 9) вычислительной структуры как R (верхняя ветвь) и D (нижняя ветвь). После проделанных преобразований (рис. 9) интервал поступления данных для блока R равен 2. Так как уменьшить длину критического пути в блоке R не представляется возможным, то для увеличения быстродействия необходимо увеличить количество регистров в обратной связи – привести обратную связь к форме конвейер в конвейере [18]. Тогда входные данные блока R смогут формироваться плотным потоком, так как данные по обратной связи будут поступать из регистров каждый такт (после их инициализации). Для того, чтобы внести в обратную связь еще один регистр, необходимо получить еще один независимый изоморфный блок R. В общем случае необходимо получить (n+m) параллельно выполняющихся изоморфных блоков с обратными связями.

Проанализировав фрагмент вычислительной структуры (рис. 9), можно заметить, что на верхней ветви формируются четные элементы выходной последовательности y (y_0, y_2, y_4 … y_n). Для того чтобы сформировать нечетные сигналы необходимо добавить после блока D блок R и образовать на нем обратную связь. Начальным значением регистра обратной связи будет выходное значение блока D. В итоге будут получены нечетные элементы выходной последовательности y (y_1, y_3, y_5 … y_{n-1}) (рис. 10).

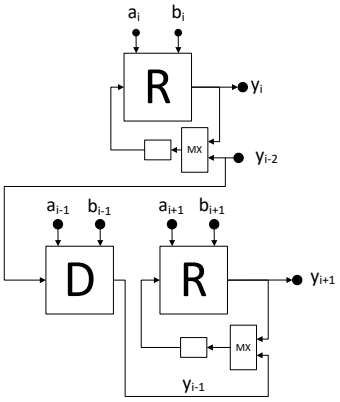


Рис. 10. Фрагмент вычислительной структуры после преобразований

Так как блоки R вычислительной структуры представляют собой независимые изоморфные структуры (подграфы), то данный фрагмент (рис. 10) может быть преобразована к форме конвейер в конвейере (рис. 11).

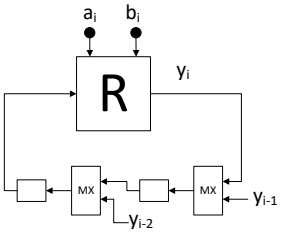


Рис. 11. Фрагмент вычислительной структуры, приведенный к форме конвейер в конвейере

В итоге, интервал поступления данных на входы блока R будет равен 1 (так как длина критического пути R равна 2, и в обратной связи находится 2 регистра). Таким образом в результате выполненных информационно-эквивалентных преобразований, удалось увеличить темп поступления входных данных (сократить интервал обработки данных в обратной связи) в 2 раза, и тем самым сократить время решения задачи в 2 раза.

Данный метод можно применять многократно, для этого к исходному фрагменту вычислительной структуры с обратной связью добавляется несколько изоморфных подграфов, общее количество которых равно исходному интервалу поступления данных. Затем над полученной структурой последовательно производятся эквивалентные преобразования дистрибутивных и ассоциативных операционных вершин.

Таким образом, применение данного метода при решении прикладных задач, в которых встречаются обратные связи с дистрибутивными вершинами, позволяет добиться плотного потока данных на входе и выходе вычислительной структуры. За счет этого удается сократить общее время решения задачи.

Расширение метода замены переменной при преобразованиях рекуррентных выражений в вычислительной структуре. Еще одним методом преобразования рекурсивных выражений является метод замены переменных [19]. Данный метод применялся только для обратных связей с интервалом обработки данных равным двум (S = 2) [19]. Рассмотрим данный метод для преобразования рекурсивных выражений на примере следующей вычислительной структуры (рис. 12):

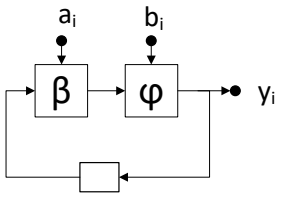


Рис. 12. Фрагмент вычислительной структуры с обратной связью

Аналогично предыдущему примеру, операционные вершины β и ϕ представляют собой любые операции, выполняемые над любыми структурами данных, для которых могут быть применены преобразования дистрибутивных и ассоциативных операционных вершин. В отличии от метода автоподстановки, данный метод требует, чтобы для любого элемента из множества K^1, в котором заданы операции выполняемые операционными вершинами β и φ, был определен обратный элемент. В общем случае описываемые далее преобразования выполняются над множеством K^1, образующим полукольцо с единицей [13, 16, 17] относительно вершин β и φ, то есть данные вершины удовлетворяют следующим условиям:

1. Для вершины φ выполняются коммуникативный и ассоциативный законы
2. Для вершины φ определен нейтральный элемент (n^0) , такой что:

$$a\ \varphi\ n^0 = n^0\ \varphi\ a = a;$$

3. Вершина β образует полугруппу с единицей относительно множества K^1, т.е. для β выполняется закон ассоциативности, а также для вершины β определен нейтральный элемент (n^1), такой что:

$$x\ \beta\ n^1 = n^1\ \beta\ x = x;$$

4. Для вершины β определен обратный элемент, такой что:

$$x\ \beta\ \bar{x} = n^1;$$

5. Вершины β и φ являются дистрибутивными относительно друг друга.

Считая, что темп обработки данных вершинами β и φ равен 1 получим следующую временную диаграмму (рис. 13), из которой видно, что данные на выходе обратной связи формируются не плотным потоком, что увеличивает время решения прикладной программы.

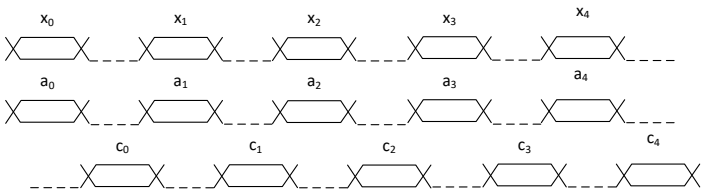


Рис. 13. Временная диаграмма обработки данных в обратной связи

Опираясь на базовые эквивалентные преобразование, преобразуем фрагмент вычислительной структуры с обратной связью (рис. 12). Воспользуемся преобразованием, которое будем называть «неполной дистрибутивностью». Суть данного преобразования заключается в использовании нейтрального и обратных элементов: выполнение операции β с нейтральным элементом (n^1) не изменяет итогового результата. При этом нейтральный элемент (n^1) может быть представлен как $n^1 = a\ \beta\ \bar{a}$. В итоге данное преобразование имеет следующий вид (рис. 14).

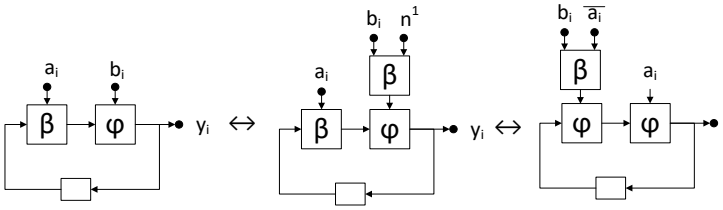
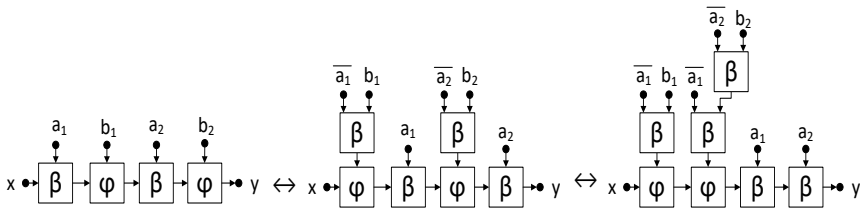


Рис. 14. Использование преобразования «неполной дистрибутивности» для пары дистрибутивных операционных вершин

В зависимости от количества итераций, которые необходимо выполнить во время вычислений, в обратной связи будет выполнено по n операций β и n операций φ, где n – общее количество итераций. Используя преобразование «неполной дистрибутивности», все операционные вершины β могут быть перемещены в одну сторону относительно операционных вершин φ. Рассмотрим действие метода более подробно, на примере отдельных итераций (рис. 15).



а

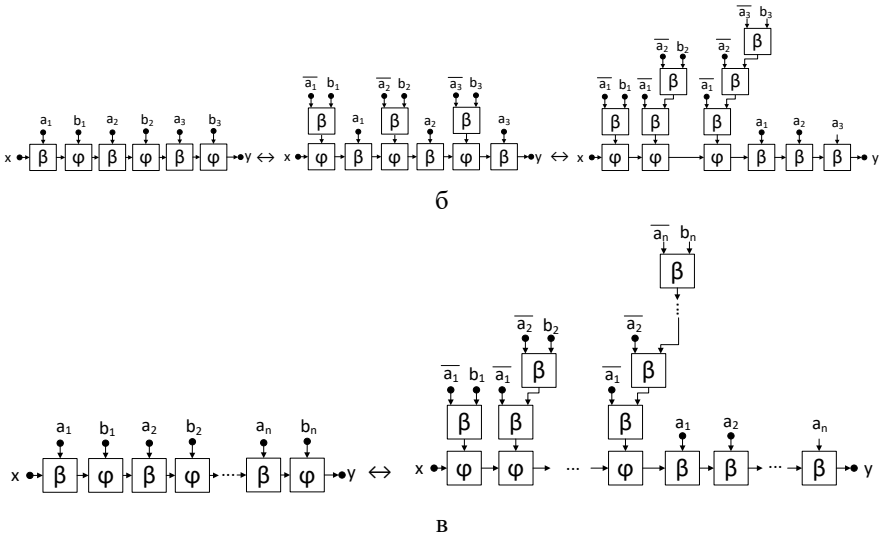


Рис. 15. Применение преобразования «неполной дистрибутивности» для последовательности дистрибутивных операционных вершин*: а – для двух пар вершин; б – для трех пар вершин; в – для n пар вершин*

Используя преобразование ассоциативных операционных вершин, преобразуем обобщенную вычислительную структуру, показанную на рис. 15,в.

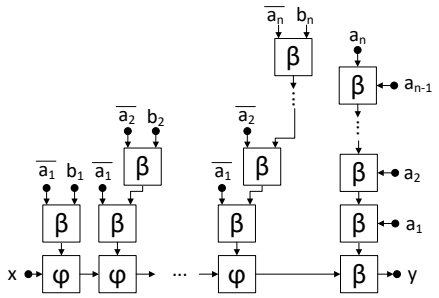


Рис. 16. Фрагмент вычислительной структуры после преобразования ассоциативных операционных вершин

Как можно заметить внешний сигнал, поступающий на операционную вершину φ (рис. 16) имеет дублирующие фрагменты. Избавимся от них, разветвив соответствующие выходные сигналы вершин β (рис. 17).

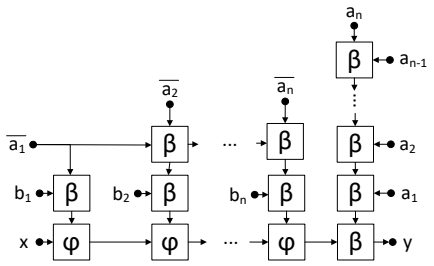


Рис. 17. Применение преобразования удаления дублирующих операционных вершин

Далее применим эквивалентное преобразование свертки/развертки последовательности ассоциативных вершин (рис. 18):

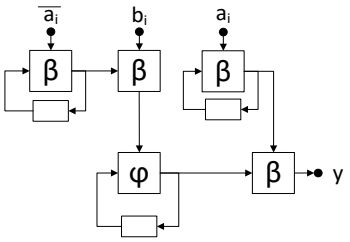


Рис. 18. Итоговый вид фрагмента вычислительной структуры после проведения всех преобразований

Используя данный метод преобразования удается избавиться от обратной связи с двумя дистрибутивными вершинами, и перейти к трем обратным связям с одной ассоциативной операционной вершиной. Если темп обработки данных вершинами β и φ равен единице, тогда удается сразу добиться плотного потока данных. Однако в случае если темп обработки данных операционными вершинами в обратной связи выше единицы, что наблюдается в современных прикладных задачах, то применение рассмотренного метода не позволяет сформировать плотный поток данных.

Для того, чтобы добиться плотного потока данных для случаев, когда темп обработки данных вершинами β и φ выше единицы, необходимо продолжить преобразования, применив метод раскрытия ассоциативных операционных вершин [20].

Обозначим «пирамидальную» часть преобразования раскрытия ассоциативных операций [20] как «Λ». Тогда фрагмент вычислительной структуры (рис. 18) может быть преобразован к следующему виду:

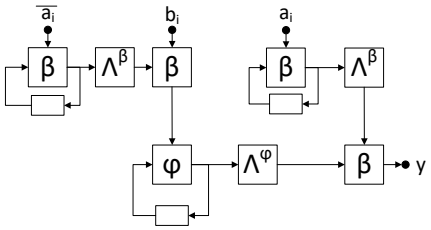


Рис. 19. Применение метода преобразования одиночных ассоциативных вершин

Использование данного преобразования позволяет добиться плотного потока данных (с интервалом обработки данных равным единице) в каждой из обратных связей без изменения порядка следования элементов в исходном потоке. При этом проведение данного преобразования требует наличия дополнительного аппаратного ресурса, что в настоящее время не является проблемой, так как современные ПЛИС имеют большой вычислительный ресурс, который продолжает расти, в отличии от количества доступных каналов информационного обмена.

Следует отметить, что, по сравнению с исходной вычислительной структурой (рис. 12), итоговая структура обладает меньшей устойчивостью. Узким местом полученной структуры являются участки обратных связей с одной ассоциативной вершиной, в которых происходит накопление результата потока данных a и $\overline{a}$, так как значение «накопления» в этом месте может переполниться или сброситься в ноль.

Применение данного метода преобразований позволит сформировать плотный поток данных на выходах обратных связей, что приведет к сокращению времени решения прикладных задач в различных предметных областях примерно в 2–5 раз.

Заключение. Предложенные методы преобразования рекурсивных выражений позволяют без участия схемотехника оптимизировать фрагменты вычислительной структуры с высоким интервалом обработки данных и сократить время решения задачи. Отличительной особенностью разработанных методов является их применение над информационно-вычислительной структурой, а не текстом исходной программы. Это позволяет проще выявлять структуры с обратными связями, и применять к ним набор базовых эквивалентных преобразований: преобразования дистрибутивных и ассоциативных операционных вершин, свертка/развертка последовательностей ассоциативных вершин, преобразования с нейтральными и обратными элементами. Базовые эквивалентные преобразования могут объединяться и образовывать методы преобразования более сложных вычислительны структур, таких как структуры с обратными связями (метод автоподстановки и метод замены переменной).

Результаты экспериментальных исследований разработанных методов в прототипе оптимизирующего синтезатора при решении задач линейной алгебры и цифровой обработки сигналов позволяют сделать вывод о сокращении времени решения прикладных задач примерно в 2–5 раз.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Гузик В.Ф., Каляев И.А., Левин И.И.* Реконфигурируемые вычислительные системы: учебное пособие / под общей ред. И.А. Каляева. – Ростов-на-Дону: Изд-во ЮФУ, 2016. – 472 с.
2. *Compton K.* Reconfigurable Computing: A Survey of Systems and Software // ACM Computing Surveys. – June 2002. – Vol. 34, No. 2. – P. 171-210.
3. *Левин И.И., Дордопуло А.И., Каляев И.А., Доронченко Ю.И., Раскладкин М.К.* Современные и перспективные высокопроизводительные вычислительные системы с реконфигурируемой архитектурой // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. – 2015. – Т. 4, № 3. – С. 24-39.
4. *Muslim F.B., Ma L., Roozmeh M. and Lavagno L*. Efficient FPGA Implementation of OpenCL High-Performance Computing Applications via High-Level Synthesis // In IEEE Access. – 2017. – Vol. 5. – P. 2747-2762. – DOI: 10.1109/ACCESS.2017.2671881.
5. *Бибило П.Н., Романов В.И.* Интеграция САПР для синтеза логических схем с использованием глобальной оптимизации // Программные продукты и системы. – 2019. – № 1. – С. 26-33.
6. Intel® Quartus® Prime Standard Edition User Guide 18.1. Getting Started. UG-20173 | 2019.12.16.
7. Xilinx Vivado Design Suite. User Guide. Synthesis. UG901 (v2019.1) June 12, 2019.
8. Synopsys Synplify Pro for Microsemi Edition User Guide November 2016.
9. Rene Mueller, Jens Teubner, and Gustavo Alonso. Data processing on FPGAs. PVLDB, 2(1), August 2009.
10. *Верещагин Н.К., Шень А.* Лекции по математической логике и теории алгоритмов. Ч. 1. Начала теории множеств. – 4-е изд., доп. – М.: МЦНМО, 2012. – 112 с.
11. *Коляда В.И., Кореновский А.А*. Курс лекций по математическому анализу: в 2-х ч. Ч. 1. – Одесса: Астропринт, 2009. – XXVII. – 369 с.
12. *Новиков Ф.А*. Дискретная математика для программистов: учебник для вузов. – 3-е изд. – СПб.: Питер, 2009. – 384 с.
13. *Воеводин В.В.* Линейная алгебра. – 2-е изд., испр. и доп. – М.: Главная редакция физико-математической литературы, 1980.
14. *Kuck D.* The structure of computers and computations. John Wiley and Sons. Inc., New York, NY, 1978.

15. *Самофалов К.Г., Луцкий Г.М.* Основы теории многоуровневых конвейерных вычислительных систем. – М.: Радио и связь, 1989. – 272 с.
16. *Васильев А.В., Мазуров В.Д.* Высшая алгебра: Конспект лекций. В 2 ч. Ч. 1. – Новосибирск: Новосиб. гос. ун-т, 2010. – 143 с.
17. *Ricky Aditya, Muhammad Taufiq Zulfikar, Ngarap Imanuel Manik.* Testing Division Rings and Fields Using a Computer Program // Procedia Computer Science. – 2015. – Vol. 59. – P. 540-549.
18. *Доронченко Ю.И.* Метод операционно-графового описания одновременных вычислений для многопроцессорных систем // Матер. Междунар. науч.-техн. конф. “Многопроцессорные вычислительные и управляющие системы – 2007”. – Таганрог: ТТИ ЮФУ, 2007. – Т. 1. – С. 11-17.
19. *Станишевский О.Б.* Эффективность арифметической обработки при конвейерных и нейроконвейерных вычислениях // Конвейерные вычислительные системы: Тезисы докладов. – Кишинев, 1988.
20. *Дудко С.А.* Эквивалентные преобразования рекуррентных выражений в реконфигурируемых вычислительных системах // Суперкомпьютерные технологии (СКТ-2020): Матер. 6-й Всероссийской научно-технической конференции. – Ростов-на-Дону, Таганрог: Изд-во ЮФУ, 2020. – Т. 1. – С. 181-183.

REFERENCES

1. *Guzik V.F., Kalyaev I.A., Levin I.I.* Rekonfiguriruemye vychislitel'nye sistemy: uchebnoe posobie [Reconfigurable computing systems: a textbook], ed. by I.A. Kalyaeva. Rostov-on-Don: Izd-vo YuFU, 2016, 472 p.
2. *Compton K.* Reconfigurable Computing: A Survey of Systems and Software, *ACM Computing Surveys*, June 2002, Vol. 34, No. 2, pp. 171-210.
3. *Levin I.I., Dordopulo A.I., Kalyaev I.A., Doronchenko Yu.I., Raskladkin M.K.* Sovremennye i perspektivnye vysokoproizvoditel'nye vychislitel'nye sistemy s rekonfiguriruemoy arkhitekturoy [Modern and promising high-performance computing systems with reconfigurable architecture], *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Vychislitel'naya matematika i informatika* [Bulletin of the South Ural State University. Series: Computational Mathematics and Computer Science], 2015, Vol. 4, No. 3, pp. 24-39.
4. *Muslim F.B., Ma L., Roozmeh M. and Lavagno L.* Efficient FPGA Implementation of OpenCL High-Performance Computing Applications via High-Level Synthesis, *In IEEE Access*, 2017, Vol. 5, pp. 2747-2762. DOI: 10.1109/ACCESS.2017.2671881.
5. *Bibilo P.N., Romanov V.I.* Integratsiya SAPR dlya sinteza logicheskikh skhem s ispol'zovaniem global'noy optimizatsii [Integration of CAD for synthesis of logic circuits using global optimization], *Programmnye produkty i sistemy* [Software products and systems], 2019, No. 1, pp. 26-33.
6. Intel® Quartus® Prime Standard Edition User Guide 18.1. Getting Started. UG-20173 | 2019.12.16.
7. Xilinx Vivado Design Suite. User Guide. Synthesis. UG901 (v2019.1) June 12, 2019.
8. Synopsys Synplify Pro for Microsemi Edition User Guide November 2016.
9. Rene Mueller, Jens Teubner, and Gustavo Alonso. Data processing on FPGAs. PVLDB, 2(1), August 2009.
10. *Vereshchagin N.K., Shen' A.* Lektsii po matematicheskoy logike i teorii algoritmov. Ch. 1. Nachala teorii mnozhestv [Lectures on mathematical logic and the theory of algorithms. Part 1. The beginnings of set theory]. 4th ed. Moscow: MTSNMO, 2012, 112 p.
11. *Kolyada V.I., Korenovskiy A.A.* Kurs lektsiy po matematicheskomu analizu [A course of lectures on mathematical analysis]: In 2 part. Part 1. Odessa: Astroprint, 2009, XXVII, 369 p.
12. *Novikov F.A.* Diskretnaya matematika dlya programmistov: uchebnik dlya vuzov [Discrete mathematics for programmers: textbook for universities]. 3rd ed. Saint Petersburg: Piter, 2009, 384 p.
13. *Voevodin V.V.* Lineynaya algebra [Linear algebra]. 2nd ed. Moscow: Glavnaya redaktsiya fiziko-matematicheskoy literatury, 1980.
14. *Kuck D.* The structure of computers and computations. John Wiley and Sons. Inc., New York, NY, 1978.
15. *Samofalov K.G., Lutskiy G.M.* Osnovy teorii mnogourovnevykh konveyernykh vychislitel'nykh system [Fundamentals of the theory of multilevel conveyor computing systems]. Moscow: Radio i svyaz', 1989, 272 p.

16. *Vasil'ev A.V., Mazurov V.D.* Vysshaya algebra: Konspekt lektsiy. V 2 ch. [Higher Algebra: Lecture Notes. In 2 part]. Novosibirsk: Novosib. gos. un-t, 2010, Part 1, 143 p.
17. *Ricky Aditya, Muhammad Taufiq Zulfikar, Ngarap Imanuel Manik*. Testing Division Rings and Fields Using a Computer Program, *Procedia Computer Science*, 2015, Vol. 59, pp. 540-549.
18. *Doronchenko Yu.I.* Metod operatsionno-grafovogo opisaniya odnovremennykh vychisleniy dlya mnogoprotsessornykh sistem [Method of operational graph description of simultaneous computations for multiprocessor systems], *Mater. Mezhdunar. nauch.-tekhn. konf. "Mnogoprotsessornye vychisl. i upravlyayushchie sistemy – 2007"* [Materials of the International Scientific and Technical Conference "Multiprocessor computing and Control systems - 2007"]. Taganrog: TTI YuFU, 2007, Vol. 1, pp. 11-17.
19. *Stanishevskiy O.B.* Effektivnost' arifmeticheskoy obrabotki pri konveyernykh i neyrokonveyernykh vychisleniyakh [The efficiency of arithmetic processing in conveyor and neuroconveyor calculations], *Konveyernye vychislitel'nye sistemy: Tezisy dokladov* [Conveyor computing systems: Abstracts]. Kishinev, 1988.
20. *Dudko S.A.* Ekvivalentnye preobrazovaniya rekurrentnykh vyrazheniy v rekonfiguriruemykh vychislitel'nykh sistemakh [Equivalent transformations of recurrent expressions in reconfigurable computing systems], *Superkomp'yuternye tekhnologii (SKT-2020): Mater. 6-y Vserossiyskoy nauchno-tekhnicheskoy konferentsii* [Supercomputer Technologies (SKT-2020): Materials of the 6th All-Russian Scientific and Technical Conference]. Rostov-on-Don, Taganrog: Izd-vo YuFU, 2020, Vol. 1, pp. 181-183.

Статью рекомендовал к опубликованию д.т.н. С.Г. Капустян.

Дудко Сергей Анатольевич – Южный федеральный университет; e-mail: dudko@sfedu.ru; г. Таганрог, Россия; тел.: +79034318173; кафедра интеллектуальных и многопроцессорных систем; аспирант.

Левин Илья Израилевич – e-mail: iilevin@sfedu.ru; тел.: 88634612111; кафедра интеллектуальных и многопроцессорных систем; зав. кафедрой; д.т.н.; профессор.

Dudko Sergei Anatolievich – Southern Federal University, e-mail: dudko@sfedu.ru; Taganrog, Russia; phone: +79034318173; the department of intellectual and multiprocessor systems; graduate student.

Levin Ilya Izrailevich – e-mail: iilevin@sfedu.ru; phone: +78634612111; head of department of intellectual and multiprocessor systems; head of department; dr. of eng. sc.; professor.

УДК 004.633 DOI 10.18522/2311-3103-2021-5-168-176

Н.К. Жуков, В.А. Мордвинов, А.А. Русляков

МОДИФИКАЦИЯ МЕТОДА ИДЕАЛЬНОЙ ТОЧКИ В НОРМИРОВАНИИ И ГАРМОНИЗАЦИИ КОНТЕНТА В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Рассмотрена разработанная авторизированная методика, основанная на Методе идеальной точки с использованием множества Парето, позволяющая с современных технологических позиций взглянуть на особенности информационного взаимодействия для оценочной деятельности и регулирования межагентных взаимодействий, которая была положена в основу предложенных при участии авторов обновлений дисциплины Российского Технологического Университета (МИРЭА), Института Информационных технологий, кафедры Инструментального и Прикладного Программного Обеспечения «Информационный менеджмент систем» четвертого курса бакалавриата направления подготовки 09.03.04 «Программная инженерия» (по профилю «Разработка программных продуктов и проектирование информационных систем»). Описаны основные преимущества Метода идеальной точки с применением множества Парето. Представлено математическое описание множества Парето и Метода идеальной точки. Применение модернизированного метода позволяет улучшить показатели эмерджентности в процессе усовершенствования