УДК 004.722.23 DOI: 10.17587/it.25.522-530

А. Ю. Романов, канд. техн. наук, доц., e-mail: a.romanov@hse.ru,
Е. А. Ведмидь, студент, e-mail: eavedmid@edu.hse.ru,
Национальный исследовательский университет "Высшая школа экономики", Москва,
Э. А. Монахова, канд. техн. наук, доц., ст. науч. сотр., e-mail: emilia@rav.sscc.ru,
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Проектирование сетей на кристалле с топологией кольцевой циркулянт с тремя образующими: разработка алгоритмов маршрутизации

Представлена реализация нескольких алгоритмов маршрутизации динамического типа, предназначенных для использования в сетях на кристалле с циркулянтной топологией типа $C(N; 1, s_2, s_3)$ для поиска кратчайших маршрутов между любыми двумя узлами сети. Разработанные алгоритмы могут быть реализованы в виде цифровых автоматов для выбора направления движения пакетов в маршрутизаторах. Проведено тестирование алгоритмов на различных наборах оптимальных циркулянтов и выполнено их сравнение по эффективности, скорости и занимаемым в памяти ресурсам.

Ключевые слова: *сеть на кристалле, алгоритм Дейкстры, кольцевой циркулянт с тремя образующими, алгоритмы маршрутизации*

Введение

Одним из важных и актуальных направлений исследований в области информатики и вычислительных систем на современном этапе является построение многоядерных процессоров [1]. При этом в условиях роста популярности технологий построения систем на кристалле (Systems on Chip, SoCs) [2] и мультипроцессорных систем на кристалле (Multi-Processor Systems on Chip, MPSoCs) [3] также возрастает и распространение сетей на кристалле (Networks-on-Chip, NoCs, СтнК) [4], которые способны решать множество проблем, присущих системам, построенным по технологии общей шины [5].

Одной из самых актуальных и важных проблем в области исследований сетей на кристалле является поиск оптимальных топологий для их построения. Стандартные регулярные топологии, такие как mesh, torus, hypercube или spidergon [6—8], не всегда способны удовлетворить современным требованиям к сетям на кристалле [9, 10]. Поэтому остро стоит вопрос поиска новых топологий, и перспективными выглядят циркулянтные топологии [11], поскольку они имеют лучшие характеристики по сравнению с классическими топологиями [12]. При этом требуется разработка простых алгоритмов маршрутизации, применимых в циркулянтных сетях.

Использовать классический алгоритм Дейкстры для маршрутизации в сетях на кристалле слишком ресурсоемко из-за большой сложности реализации алгоритма на уровне маршрутизатора СтнК или IP-ядра [13, 14]. При табличной же маршрутизации необходимо хранить всю маршрутную информацию на уровне каждого маршрутизатора, что также является ресурсозатратным решением [2]. Поэтому задачей данной работы является разработка простых алгоритмов различных типов, которые могут быть реализованы в виде RTL цифровых автоматов на уровне маршрутизаторов в СтнК.

1. Кольцевые циркулянты с тремя образующими

Циркулянтные графы вида $C(N; 1, s_2, s_3)$, где $2 \leqslant s_2 < s_3 < N$, называются кольцевыми циркулянтами с тремя образующими, являющимися частным случаем кольцевых графов [15, 16]. Пример такого графа приведен на рис. 1 (см. вторую сторону обложки).

Циркулянты с тремя образующими являются перспективной топологией для проектирования сетей на кристалле. Так же, как и для топологий 3D-mesh и 3D-torus [12], маршрутизаторы в таких сетях содержат по шесть внешних портов, но представление таких топологий возможно в двумерном виде, что лучше всего

подходит для современных ASIC и ПЛИС, выполняемых по планарной технологии. Кроме того, циркулянты обладают лучшими характеристиками диаметра и среднего расстояния между узлами по сравнению с классическими регулярными топологиями [17].

1.1. Разработка структуры пакета при статической маршрутизации в сетях с топологией на основе кольцевых циркулянтов с тремя образующими

В большинстве сетей на кристалле используется парная маршрутизация [18], когда пакет передается из маршрутизатора источника данных в маршрутизатор узла назначения. Для организации такой маршрутизации можно воспользоваться любым алгоритмом поиска кратчайшего пути, например алгоритмом Дейкстры [16, 19]. Обычно используют статический тип маршрутизации [2], где каждый маршрутизатор каждого узла хранит список, каждый элемент которого является одним из узлов сети и представляет собой другой список с номерами узлов, с которыми соединен данный узел. При этом каждый маршрутизатор знает свой порядковый номер. На вход маршрутизатору, который должен будет передать пакет с данными, поступает номер узла назначения (маршрутизатора приемника). Маршрутизатор знает структуру сети и поэтому может рассчитать кратчайший путь по одному из алгоритмов.

Суть алгоритма Дейкстры [18] заключается в следующем: каждой вершине сопоставляется метка, которая содержит минимальное известное расстояние от данной вершины до вершины A (если расстояние неизвестно, то оно считается равным бесконечности или достаточно большому числу, чтобы можно было считать его бесконечно большим). Алгоритм пошагово перебирает каждую вершину и проверяет, можно ли с помощью этой вершины (с помощью пути от стартовой вершины до текущей) уменьшить расстояние (метку) от начального узла до вершины-соседа. Работа алгоритма Дейкстры длится до тех пор, пока все вершины не будут посещены.

Алгоритм Дейкстры достаточно универсален и подходит для любых графов, на основе которых и построены циркулянты, а следовательно, он подойдет для любых типов циркулянтов, включая кольцевые с любым числом и значением образующих [13]. Основная проблема данного алгоритма заключается в том, что при увеличении числа узлов время работы и потребляемая память значительно увеличиваются. Поэтому существует необходимость в разработке специализированного алгоритма, который позволял бы маршрутизаторам рассчитывать следующий шаг пакета в сети на основе его маршрутной информации. В литературе довольно мало работ, посвященных поиску кратчайших путей в циркулянтах с тремя образующими. Существуют решения [20, 21] для отдельных семейств циркулянтов с порядком $N = O(3d^2)$, где d — диаметр. Также в работе [22] представлен простой аналитический метод поиска кратчайшего пути в циркулянтах максимального порядка при заданном диаметре. Универсального алгоритма маршрутизации для циркулянтов с тремя образующими нет, так же как и для семейства кольцевых циркулянтов.

1.2. Разработка специализированных алгоритмов маршрутизации для сетей на кристалле на основе топологии кольцевой циркулянт с тремя образующими

Число портов маршрутизатора определяется степенью вершин циркулянта как $p = 2k$, где k — размерность графа (число его образующих) [12]. Таким образом, маршрутизатор циркулянта вида $C(N; 1, s_2, s_3)$ имеет шесть соединений с другими маршрутизаторами.

Самым очевидным алгоритмом для навигации в циркулянтных сетях можно считать алгоритм табличной маршрутизации, описанный в работе [16]. Таблица маршрутизации представляет собой квадратную матрицу $N \times N$, где N — число узлов (маршрутизаторов), а ячейки содержат номера портов, в которые нужно отправить пакет, чтобы он достиг узла назначения. Каждый маршрутизатор хранит только свою строку из таблицы.

В работе [16] количество памяти, которое занимает такая таблица, описано с помощью функции

$$M = N^2 \lceil \log_2 p \rceil, \qquad (1)$$

где N — число узлов в сети; $\lceil \log_2 p \rceil$ — необходимое количество памяти в битах для хранения номеров портов маршрутизатора; p — число портов маршрутизатора.

С одной стороны, использование алгоритма табличной маршрутизации требует хранения больших объемов данных, а с другой — реализация такого алгоритма в виде цифрового автомата не занимает много места на кристалле и является довольно простой.

1.3. Алгоритм обхода графа по часовой стрелке

Для навигации в циркулянтных сетях можно использовать алгоритм, который основан на итерационном вычислении маршрута между узлами, при котором каждый маршрутизатор принимает решение о коммутации пакета в следующий маршрутизатор только на один шаг. Поскольку циркулянты симметричны, для любого узла не важен его порядковый номер, а только расстояние в хопах (переходах) от него к другим узлам. Поэтому для уменьшения размера адресного поля в пакете в качестве адреса передается разница между номерами узлов (источника данных и приемника). Нагрузка на пакет (размер адресного поля в битах) может быть вычислена по следующей формуле:

$$P = \lceil \log_2 N \rceil, \quad (2)$$

где N — число узлов в сети.

Дополнительно в маршрутизаторе необходимо хранить число узлов и значения образующих s_2 и s_3. Таким образом, общий размер хранимых данных может быть вычислен по следующей формуле:

$$M = N\left(\lceil \log_2 N \rceil + \left\lceil \log_2 \frac{N}{2} \right\rceil + \left\lceil \log_2 \left(\frac{N}{2} - 1 \right) \right\rceil\right), \quad (3)$$

где N — число узлов в сети; $\lceil \log_2 N \rceil$ — необходимое количество памяти в битах для хранения числа маршрутизаторов в сети; $\left\lceil \log_2 \frac{N}{2} \right\rceil$ — необходимое количество памяти в битах для хранения образующей s_3; $\left\lceil \log_2 \left(\frac{N}{2} - 1 \right) \right\rceil$ — необходимое количество памяти в битах для хранения образующей s_2.

Для хранения значения образующей s_3 необходимо $\left\lceil \log_2 \frac{N}{2} \right\rceil$ бит, так как образующая s_3 гарантированно будет меньше, чем половина числа узлов [3], а образующая s_2 будет как минимум на единицу меньше, чем s_3.

Вычисление перехода происходит следующим образом: сначала определяется направление перехода (по направлению или против направления движения часовой стрелки), затем происходит выбор образующей. Если разница между узлом-источником и узлом-приемником меньше, чем половина числа узлов, то выбирается движение в направлении движения часовой стрелки, если больше — то в противоположном направлении. Если выбрано направление движения по часовой стрелке, то выбор образующей происходит следующим образом:

- пока разница между узлом-источником и узлом-приемником больше значения s_3, переход будет происходить по бóльшей образующей;
- если больше s_2, но меньше s_3, то переход будет осуществляться по образующей s_2, в противном случае — по образующей $s_1 = 1$.

Значение адресного поля головного флита пересчитывается путем вычитания длины образующей, по которой будет выполнен переход. Равенство нулю значения адресного поля головного флита является критерием окончания передачи пакета. Если было выбрано движение против часовой стрелки, общий алгоритм выбора текущего шага такой же, но сравнение происходит между образующими и разницей значения числа узлов в сети со значением, хранящимся в адресном поле головного флита. Перед переходом к адресному значению в головном флите прибавляется значение длины образующей, по которой произойдет переход. Критерием окончания передачи пакета в данном случае является равенство значения адресного поля головного флита числу узлов в сети. Предложенный алгоритм имеет много общего с похожим алгоритмом для двумерных кольцевых циркулянтов, описанным в работе [23].

Описание предложенного алгоритма приведено ниже:

algorithm Find_route_Clockwise is
Input: *startNode* — start node, *endNode* — end node, N — count of nodes, s_1 — first generator, s_2 — second generator, s_3 — third generator.
Output: *startNode* — next start node.

```
1:  S ← endNode − startNode
2:  If S = 0 then
3:     return startNode
4:  If S < 0 then
5:     S ← S + N
6:  If S ⩽ N/2 then
7:     If S ⩾ s3 then
8:        startNode ← (s3 + startNode) modN
9:     else
10:       If S ⩾ s2 then
11:          startNode ← (s2 + startNode) modN
12:       else
13:          startNode ← (s1 + startNode) modN
14: else
15:    S ← N − S
16:    If S ⩾ s3 then
17:       startNode ← (N − s3 + startNode) modN
18:    else
19:       If S ⩾ s2 then
20:          startNode ← (N − s2 + startNode) modN
21:       else
22:          startNode ← (N − s1 + startNode) modN
23: If startNode = 0 then
24:    startNode ← N
25: return startNode
```

Представленный алгоритм не является оптимальным, так как в некоторых случаях он будет предлагать пути, длина которых в хопах больше диаметра сети, но зато будет существенно экономить занимаемую маршрутизатором память.

Можно немного оптимизировать данный алгоритм следующим образом: проводить сравнение разности источника и приемника с $\frac{s_3+s_2}{2}$ и $\frac{s_1+s_2}{2}$. Если разность больше первого значения, то переход будет осуществляться по образующей s_3, если разность находится между этими значениями, то по s_2, иначе — по s_1.

Общий размер хранимых данных для этого алгоритма будет равен базовому (3). Данный алгоритм работает все еще недостаточно эффективно.

1.4. Алгоритм выбора направлений

Развитием предложенного подхода является алгоритм выбора направлений, который может менять направление движения при вычислении перехода в следующий узел по аналогии с тем, как делается в работе [23]. В данном алгоритме предлагается в качестве адреса в головном флите хранить номер узла назначения. Нагрузка на пакет остается той же, что и в алгоритме обхода графа по часовой стрелке (2). Дополнительно в маршрутизаторе необходимо хранить его номер, число узлов в сети и длину образующих s_2 и s_3. Общий размер хранимых данных вычисляется по формуле

$$M = N\left(2\lceil\log_2 N\rceil + \left\lceil\log_2\frac{N}{2}\right\rceil + \left\lceil\log_2\left(\frac{N}{2}-1\right)\right\rceil\right), \quad (4)$$

где N — число узлов в сети; $\lceil\log_2 N\rceil$ — необходимое количество памяти в битах для хранения номера маршрутизатора и числа маршрутизаторов в сети; $\left\lceil\log_2\frac{N}{2}\right\rceil$ — необходимое количество памяти в битах для хранения образующей s_3; $\left\lceil\log_2\left(\frac{N}{2}-1\right)\right\rceil$ — необходимое количество памяти в битах для хранения образующей s_2.

Работу алгоритма логически можно разделить на две части. В первой части происходит выбор последовательности передачи номеров узлов источника и приемника пакета, которые передаются во вторую часть алгоритма. Это возможно из-за того, что граф является неориентированным и вершинно-транзитивным [12]. Данная процедура требуется для упрощения алгоритма расчета направления движения из-за того, что он будет работать только с положительными числами. Кроме того, в первой части алгоритма происходит нормализация полученного направления движения пакета. Во второй части алгоритма непосредственно происходит вычисление следующего шага движения пакета. Алгоритмическое описание приведено ниже.

algorithm Find_Route_Selection is
Input: *startNode* — start node, *endNode* — end node, N — count of nodes, s_1 — first generator, s_2 — second generator, s_3 — third generator.
Output: startNode — next start node.
1: If star*tNode* > *endNode* **then**
2: *startNode* ← *startNode* − *Step*(*endNode*, *startNode*, N, s_1, s_2, s_3)
3: **else**
4: *startNode* ← *startNode* + *Step*(*endNode*, *startNode*, N, s_1, s_2, s_3)
5: **If** *startNode* > N **then**
6: *startNode* ← *startNode* − N
7: **else**
8: **If** *startNode* ⩽ 0 **then**
9: *startNode* ← *startNode* + N
10: **return** ***startNode***

function Step is
Input: *startNode* — start node, *endNode* — end node, N — count of nodes, s_1 — first generator, s_2 — second generator, s_3 — third generator.
Output: the function returns the best step (direction is also selected)
1: $bestWayR \leftarrow 0$, $stepR \leftarrow 0$, $bestWayL \leftarrow 0$, $S \leftarrow endNode - startNode$
2: $R_1 \leftarrow \frac{S-1}{s_2} + S \bmod N$, $R_2 \leftarrow \frac{S-1}{s_2} - S \bmod s_2 + s_2 + 1$
3: $R_3 \leftarrow \frac{S-1}{s_2} - \left(s_2 * \left(\frac{S}{s_2}+1\right) - S\right) + s_2 + 1$,
$R_4 \leftarrow \frac{S-1}{s_3} - S \bmod s_3 + s_3 + 1$
4: $R_5 \leftarrow \frac{S-1}{s_3} - \left(s_3 * \left(\frac{S}{s_3}+1\right) - S\right) + s_3 + 1$
5: **If** $R_3 > R_2$ **then**
6: $R_3 > R_2$
7: **If** $R_5 > R_4$ **then**
8: $R_5 > R_4$
9: **If** $R_1 < R_3$ **then**
10: **If** $R_1 < R_5$ **then**
11: $bestWayR \leftarrow R_1$, $stepR \leftarrow s_1$
12: **else**
13: $bestWayR \leftarrow R_5$, $stepR \leftarrow s_3$
14: **else**
15: **If** $R_3 < R_5$ **then**
16: $bestWayR \leftarrow R_3$, $stepR \leftarrow s_2$
17: **else**
18: $bestWayR \leftarrow R_5$, $stepR \leftarrow s_3$
19: $S \leftarrow endNode - startNode + N$
20: $L_1 \leftarrow \frac{S-1}{s_2} + S \bmod N$, $L_2 \leftarrow \frac{S-1}{s_2} - S \bmod s_2 + s_2 + 1$
21: $L_3 \leftarrow \frac{S-1}{s_2} - \left(s_2 * \left(\frac{S}{s_2}+1\right) - S\right) + s_2 + 1$,
$L_4 \leftarrow \frac{S-1}{s_3} - S \bmod s_3 + s_3 + 1$

22: $L_5 \leftarrow \frac{S-1}{s_3} - \left(s_3 * \left(\frac{S}{s_3} + 1 \right) - S \right) + s_3 + 1$
23: **If** $L_3 > L_2$ **then**
24: $L_3 > L_2$
25: **If** $L_5 > L_4$ **then**
26: $L_5 > L_4$
27: **If** $L_1 < L_3$ **then**
28: **If** $L_1 < L_5$ **then**
29: $bestWayL \leftarrow L_1$, $stepL \leftarrow -s_1$
30: **else**
31: $bestWayL \leftarrow L_5$, $stepL \leftarrow -s_3$
32: **else**
33: **If** $L_3 < L_5$ **then**
34: $bestWayL \leftarrow L_3$, $stepL \leftarrow -$s2
35: **else**
36: $bestWayL \leftarrow L_5$, $stepL \leftarrow -s_3$
37: **If** $bestWayR < bestWayL$ **then**
38: **return stepR**
39: **else**
40: **return stepL**

В предложенном алгоритме учтены циклы и ситуации, когда выгоднее сначала перейти по старшей образующей, а потом вернуться по средней. Тем не менее при проверке алгоритма на графах оказалось, что он все равно не в состоянии всегда гарантировать, что пакет будет двигаться по кратчайшему пути между узлами.

Поэтому был разработан еще один вариант алгоритма, принцип работы которого приближен к алгоритму Дейкстры, но учитывает особенности рассматриваемых циркулянтов [19].

1.5. Алгоритм поиска коэффициентов при образующих графа

Можно представить поиск кратчайшего пути для топологии, основанной на кольцевом циркулянте, как следующую оптимизационную задачу:

$$Nk + s = a_1 + a_2 s_2 + a_3 s_3, \quad (5)$$

где N — число узлов в сети; k — номер рассматриваемого цикла (может быть отрицательным); s — длина пути от узла-источника к узлу-приемнику; s_2, s_3 _ образующие; a_1, a_2, a_3 — коэффициенты при образующих s_1, s_2, s_3 соответственно.

Задача оптимизации заключается в минимизации суммы абсолютных величин коэффициентов a_1, a_2, a_3. Если выразить все переменные через переменную a_1, получится уравнение, где остается три неизвестных — a_2, a_3, k:

$$a_1 = a_2 s_2 + a_3 s_3 - Nk - s. \quad (6)$$

Далее нужно выбрать, в каких границах будут изменяться переменные a_2, a_3, k, и с помощью простого циклического перебора находить значение a_1, после чего выбирать тот набор коэффициентов, сумма которых наименьшая.

Общий размер хранимых данных для такого алгоритма вычисляется по формуле

$$M = N\left(2\lceil \log_2 N \rceil + \left\lceil \log_2 \frac{N}{2} \right\rceil + \right.$$
$$\left. + \left\lceil \log_2 \left(\frac{N}{2} - 1 \right) \right\rceil + \lceil \log_2 \alpha \rceil + \lceil \log_2 \beta \rceil + \lceil \log_2 \tau \rceil \right), \quad (7)$$

где N — число узлов в сети; $\lceil \log_2 N \rceil$ — необходимое количество памяти в битах для хранения номера маршрутизатора и числа маршрутизаторов в сети; $\left\lceil \log_2 \frac{N}{2} \right\rceil$ — необходимое количество памяти в битах для хранения образующей s_3; $\left\lceil \log_2 \left(\frac{N}{2} - 1 \right) \right\rceil$ — необходимое количество памяти в битах для хранения образующей s_2; α, β, τ — коэффициенты, отвечающие за число циклов и образующих, которые будут рассматриваться в алгоритме, соответственно; $\lceil \log_2 \alpha \rceil + \lceil \log_2 \beta \rceil + \lceil \log_2 \tau \rceil$ — необходимое количество памяти в битах для хранения коэффициентов α, β, τ.

Коэффициент τ задается вручную как число циклов, которые могут быть пройдены в обе стороны. Тогда коэффициенты $\alpha = \left\lceil \frac{\tau N}{s_3} \right\rceil$ и $\beta = \left\lceil \frac{\tau N}{s_2} \right\rceil$.

Алгоритмическое описание данного варианта приведено ниже.

function Find_route_coefficients is
Input: *startNode* — start node, *endNode* — end node, N — count of nodes, s_1 — first generator, s_2 — second generator, s_3 — third generator.
Output: the function returns the best coefficients a_1, a_2, a_3
1: $bestA1 \leftarrow$ maxint, $bestA2 \leftarrow$ maxint, $bestA3 \leftarrow$ maxint
2: $S \leftarrow endNode - startNode$, $a1 \leftarrow 0$
3: $zero \leftarrow 10$, $alpha \leftarrow (zero*N)divs_3$, $beta \leftarrow (zero*N)divs_2$
4: **For all** $k \in (-zero, zero)$:
5: **For all** $a_3 \in (-alpha, alpha)$:
6: **For all** $a_2 \in (-beta, beta)$:
7: $a_1 \leftarrow k*N + S - a_3*s_3 - a_2*s_2$
8: **If** $|a_1| + |a_2| + |a_3| < |bestA1| + |bestA2| + |bestA3|$ **then**
9: $bestA1 \leftarrow a1$, $bestA2 \leftarrow a2$, $bestA3 \leftarrow a3$
10: **return** $bestA1$, $bestA2$, $bestA3$

2. Тестирование предложенных алгоритмов

Для оценки работы алгоритмов предлагается использовать критерий эффективности, учитывающий число требуемых шагов для

прохождения пакетом из первого узла во все остальные [18]. В качестве оптимального алгоритма принят алгоритм Дейкстры [19], который гарантированно находит кратчайшие пути между всеми узлами любого связного графа. Следовательно, критерий эффективности определяется формулой

$$K = \frac{\sum_{i=1}^{N-1} H^{D}_{(0-i)}}{\sum_{i=1}^{N-1} H^{A}_{(0-i)}}, \qquad (8)$$

где N — число узлов в сети; $\sum_{i=1}^{N-1} H^{A}_{(0-i)}$ — число хопов из нулевого узла во все остальные, рассчитанное по используемому алгоритму; $\sum_{i=1}^{N-1} H^{D}_{(0-i)}$ — число хопов из нулевого узла во все остальные, рассчитанное по алгоритму Дейкстры.

Для тестирования алгоритмов выбраны кольцевые циркулянты вида $C(N; 1, s_2, s_3)$, где N — число узлов в сети — определяется формулой $N = n^2$ при $N \leqslant 100$ и 150, 200, 300, 400, 500 (n — натуральное число, чтобы можно было сравнить результаты работы сетей с топологией кольцевого циркулянта с тремя образующими с топологиями torus и mesh). Тестируемые циркулянты являются оптимальными и имеют минимальный диаметр среди кольцевых циркулянтов с таким же числом узлов [16]. Результаты тестирования приведены в табл. 1.

Для алгоритмов Дейкстры и табличной маршрутизации коэффициент эффективности будет всегда равен 1, так как первый из них является эталонным, а второй предполагает наличие оптимального пути. Для алгоритма обхода по часовой стрелке эффективность сильно зависит от большей образующей и разности между образующими s_2 и s_3 — чем они больше, тем меньше эффективность алгоритма. Для алгоритма поиска коэффициентов эффективность оказалась равна 1 для всех циркулянтов.

В табл. 2 представлены результаты сравнения разработанных алгоритмов маршрутизации по длине максимального пути в графе.

Таким образом, для алгоритма обхода по часовой стрелке и его улучшенной версии полученный максимальный путь в графе отличается значительно в худшую сторону от аналогичного показателя, полученного с помощью алгоритма Дейкстры (для некоторых случаев в несколько раз). Алгоритм выбора направлений показывает лучшее качество, но в некото-

Таблица 1

Сравнение эффективности разработанных алгоритмов

Циркулянт	Алгоритм			
	Обход по часовой стрелке	Улучшенный обход по часовой стрелке	Выбор направлений	Поиск коэффициентов
C(9; 1, 2, 4)	1	1	1	1
C(16; 1, 4, 8)	0,818	1	1	1
C(25; 1, 6, 10)	0,742	0,821	0,939	1
C(36; 1, 8, 15)	0,656	0,687	0,955	1
C(49; 1, 10, 23)	0,527	0,685	0,887	1
C(64; 1, 12, 30)	0,481	0,643	0,909	1
C(81; 1, 15, 37)	0,474	0,646	0,959	1
C(100; 1, 17, 40)	0,441	0,588	0,781	1
C(100; 1, 10, 30)	0,689	1	1	1
C(150; 1, 33, 59)	0,329	0,536	0,795	1
C(200; 1, 56, 87)	0,291	0,525	0,804	1
C(300; 1, 74, 138)	0,148	0,279	0,837	1
C(400; 1, 69, 195)	0,195	0,321	0,968	1
C(400; 1, 65, 199)	0,342	0,368	0,988	1
C(500; 1, 34, 200)	0,537	0,947	0,968	1

Таблица 2

Сравнение максимальных путей в графе для разработанных алгоритмов

Циркулянт	Алгоритм				
	Дейкстры	Обход по часовой стрелке	Улучшенный обход по часовой стрелке	Выбор направлений	Поиск коэффициентов
C(9; 1, 3, 5)	2	2	2	2	2
C(16; 1, 4, 8)	3	4	3	3	3
C(25; 1, 6, 10)	3	5	4	3	3
C(36; 1, 8, 15)	4	7	5	5	4
C(49; 1, 10, 23)	4	10	6	5	4
C(64; 1, 12, 30)	4	12	7	6	4
C(81; 1, 15, 37)	5	15	9	6	5
C(100; 1, 17, 40)	6	17	10	9	6
C(100; 1, 10, 30)	7	11	7	7	7
C(150; 1, 33, 59)	8	32	17	11	8
C(200; 1, 56, 87)	12	55	28	15	12
C(300; 1, 74, 138)	8	73	37	10	8
C(400; 1, 69, 195)	11	66	36	13	11
C(400; 1, 65, 199)	9	66	34	23	9
C(500; 1, 34, 200)	18	37	19	20	18

рых случаях различие между значениями максимального пути графа достигает 14. Алгоритм поиска коэффициентов показывает такой же результат, как и эталонный алгоритм.

Работа алгоритма поиска коэффициентов была проверена на 502 оптимальных графах из датасета [24], полученных с помощью программного обеспечения, описанного в работе [16], в результате чего подтверждено, что при заданных коэффициентах α, β, τ, равных 10, 20 и 30 соответственно, эффективность алгоритма не опускается ниже 1.

Также была проведена проверка алгоритма выбора направлений на зависимость эффективности алгоритма от разницы между образующими s_2 и s_3. В результате тестирования алгоритма на данных работы [24], среди которых было 502 оптимальных кольцевых циркулянта, было получено, что при увеличении разницы между образующими s_2, s_3 прослеживается тенденция к снижению эффективности алгоритма.

На рис. 2 (см. вторую сторону обложки) представлен график зависимости между эффективностью алгоритма выбора направлений и разницей между его старшими образующими.

Исходя из приведенных выше формул (3), (4), (7) расчет памяти в битах, необходимой для работы алгоритма, приведен в табл. 3.

Рассматриваемые алгоритмы реализованы на языке программирования Python 3, что позволило оценить время их работы для поиска всех путей в кольцевом графе. Тестирование проведено на компьютере с операционной системой Windows 8.1, оперативной памятью 12 Гб и четырехъядерным процессором с частотой 2,4 ГГц. Алгоритм табличной маршрутизации не рассматривался, так как он не подразумевает сложных вычислений. Результаты алгоритма обхода по часовой стрелке практически не отличаются от результатов улучшенного алгоритма обхода по часовой стрелке и поэтому объединены в один столбец. Полученные показатели времени работы алгоритмов в миллисекундах представлены в табл. 4.

Заключение

Проведено исследование использования циркулянтных топологий размерности три для проектирования сетей на кристалле. Для кольцевых циркулянтов с тремя образующими разработан ряд алгоритмов парной маршрутизации: алгоритм на основе табличной маршру-

Таблица 3

Занимаемые ресурсы памяти в битах в зависимости от типа алгоритма

Циркулянт	Алгоритм			
	Табличная маршрутизация	Улучшенный обход по часовой стрелке	Выбор направлений	Поиск коэффициентов
C(9; 1, 3, 5)	243	108	144	270
C(16; 1, 4, 8)	768	192	256	480
C(25; 1, 6, 10)	1875	375	500	850
C(36; 1, 8, 15)	3888	648	864	1368
C(49; 1, 10, 23)	7203	882	1176	1862
C(64; 1, 12, 30)	12 288	1152	1536	2432
C(81; 1, 15, 37)	19 683	1701	2268	3402
C(100; 1, 17, 40)	30 000	2100	2800	4200
C(150; 1, 33, 59)	67 500	3600	4800	6900
C(200; 1, 56, 87)	120 000	4800	6400	9200
C(300; 1, 74, 138)	270 000	8100	10 800	15 000
C(400; 1, 65, 199)	480 000	10 800	14 400	20 000
C(500; 1, 34, 200)	750 000	13 500	18 000	25 000

Примечание: при подсчете памяти для алгоритма поиска коэффициентов константы α, β, τ выбраны равными 10, 20, 30 соответственно

Таблица 4

Время работы алгоритмов, мс

Циркулянт	Алгоритм		
	Улучшенного обхода по часовой стрелке	Выбора направлений	Поиска коэффициентов
C(9; 1, 3, 5)	0,003504	0,006079	28,323078
C(16; 1, 4, 8)	0,003981	0,012421	50,129890
C(25; 1, 6, 10)	0,005578	0,019598	96,979403
C(36; 1, 8, 15)	0,010770	0,036001	116,926932
C(49; 1, 10, 23)	0,018406	0,054717	153,540992
C(64; 1, 12, 30)	0,037407	0,080180	202,219200
C(81; 1, 15, 37)	0,044202	0,113415	261,775398
C(100; 1, 17, 40)	0,090599	0,269699	328,511905
C(150; 1, 33, 59)	0,298190	0,450205	510,957384
C(200; 1, 56, 87)	0,491786	0,728797	656,596207
C(300; 1, 74, 138)	0,867891	0,984096	994,344091
C(400; 1, 65, 199)	0,913595	2,111387	1324,184012
C(500; 1, 34, 200)	0,977516	2,547621	1682,586193

тизации, алгоритм обхода по часовой стрелке, алгоритм выбора направлений и оптимальный алгоритм, основанный на поиске коэффициентов. Показано, что классический алгоритм с табличной маршрутизацией в сетях на кристалле может быть заменен на алгоритм поиска коэффициентов при образующих, так как он обеспечивает такое же число хопов между узлами и является оптимальным, при этом его реализация на аппаратном уровне требует значительно меньше ресурсов памяти. Также альтернативными вариантами при низких требованиях к пропускной способности СтнК, но в условиях ограниченности аппаратных ресурсов, могут быть алгоритм обхода по часовой стрелке и его улучшенная версия, а также алгоритм выбора направлений, использование которых позволяет почти в два раза снизить затраты аппаратных ресурсов, но приводит к значительному увеличению длины максимального пути в графе и среднего расстояния между узлам.

Проведено сравнение сложности алгоритмов, а также ресурсов, занимаемых синтезированными подсистемами связи СтнК в ПЛИС. Поскольку предлагаемые алгоритмы, в отличие от классического алгоритма Дейкстры, не требуют рассчитывать весь путь прохождения пакета, а определяют только номер порта для следующего шага, гарантируя, что пакет достигнет узла назначения, они могут быть легко реализованы в виде цифрового автомата в маршрутизаторах для сетей на кристалле.

Публикация подготовлена в ходе проведения исследования (№ 18-01-0074) в рамках Программы "Научный фонд Национального исследовательского университета "Высшая школа экономики" (НИУ ВШЭ)" в 2018—2019 гг. и в рамках государственной поддержки ведущих университетов Российской Федерации "5-100".

Исследование Монаховой Э. А. выполнено в рамках проекта ИВМиМГ СО РАН 0315-2016-0006.

Список литературы

1. **Dahir N. S. et al.** Modeling and tools for power supply variations analysis in networks-on-chip // IEEE Trans. Comput. 2014. Vol. 63, N. 3. P. 679—690.

2. **Benmessaoud Gabis A., Koudil M.** NoC routing protocols — objective-based classification // J. Syst. Archit. Elsevier B. V., 2016. Vol. 66—67. P. 14—32.

3. **Paul J. et al.** Resource-awareness on heterogeneous MPSoCs for image processing // J. Syst. Archit. 2015. Vol. 61, N. 10. P. 668—680.

4. **Abdelfattah M. S., Bitar A., Betz V.** Design and Applications for Embedded Networks-on-Chip on FPGAs // IEEE Trans. Comput. 2017. Vol. 66, N. 6. P. 1008—1021.

5. **Angiolini F. et al.** A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs // IEEE Trans. Comput. Des. Integr. Circuits Syst. 2007. Vol. 26, N. 3. P. 421—434.

6. **Marvasti M. B., Szymanski T. H.** The performance of hypermesh NoCs in FPGAs // IEEE International Conference on Computer Design: VLSI in Computers and Processors. 2012. P. 492—493.

7. **Bishnoi R. et al.** Distributed adaptive routing for spidergon NoC // 18th International Symposium on VLSI Design and Test, VDAT 2014. 2014. P. 1—6.

8. **Deb D. et al.** Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines // J. Parallel Distrib. Comput. Academic Press, 2019. Vol. 123. P. 118—129.

9. **Dally W. J., Towles B. P.** Principles and Practices of Interconnection Networks. Elsevier, 2003. 581 p.

10. **Kumar A., Tyagi S., Jha C. K.** Performance analysis of network-on-chip topologies // J. Inf. Optim. Sci. 2017. Vol. 38, N. 6. P. 989—997.

11. **Vilfred V.** A few properties of circulant graphs: Self-complementary, isomorphism, Cartesian product and factorization // 2017 7th International Conference on Modeling, Simulation, and Applied Optimization, ICMSAO 2017. 2017. P. 1—5.

12. **Монахова Э. А.** Структурные и коммуникативные свойства циркулянтных сетей // Прикладная дискретная математика. 2011. Т. 3, № 13. С. 92—115.

13. **Cai J. Y. et al.** On routing in circulant graphs // Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 1999. Vol. 1627. P. 360—369.

14. **Кузнецов Н. А., Фетисов В. Н.** Алгоритм Дейкстры с улучшенной робастностью для управления маршрутизацией в IP-сетях // Автоматика и телемеханика. 2008. Т. 2. С. 80—85.

15. **Монахова Э. А.** Мультипликативные циркулянтные сети // Дискретный анализ и исследование операций. 2010. Т. 17. С. 56—66.

16. **Romanov A. Yu., Romanova I. I., Glukhikh A. Yu.** Development of a Universal Adaptive Fast Algorithm for the Synthesis of Circulant Topologies for Networks-on-Chip Implementations // 2018 IEEE 38th International Scientific Conference on Electronics and Nanotechnology, ELNANO 2018. 2018. P. 110—115.

17. **Монахова Э. А., Монахов О. Г.** О некоторых характеристиках циркулянтных и тороидальных структур вычислительных систем // Вестник СибГУТИ. 2013. № 3. С. 63—69.

18. **Dijkstra E. W.** A note on two problems in connexion with graphs // Numer. Math. 1959. Vol. 1. P. 269—271.

19. **Левитин А. В.** Жадные методы: Алгоритм Дейкстры // Алгоритмы. Введение в разработку и анализ. 2006. С. 189—195.

20. **Barrière L. et al.** Fault-tolerant routings in chordal ring networks // Networks. 2000. Vol. 36, N. 3. P. 180—190.

21. **Liestman A. L., Opatrny J., Zaragoza M.** Network properties of double and triple fixed step graphs // Int. J. Found. Comput. Sci. 1998. Vol. 9, N. 1. P. 57—76.

22. **Monakhova E. A.** Optimal Triple Loop Networks with Given Transmission Delay: Topological Design and Routing // International Network Optimization Conference. Paris, 2003. P. 410—415.

23. **Romanov A. Yu.** Development of routing algorithms in networks-on-chip based on ring circulant topologies // Heliyon. Elsevier, 2019. Vol. 5, N. 4. P. e01516.

24. **Romanov A. Yu.** Optimal Circulants Dataset [Electronic resource]. URL: https://github.com/RomeoMe5/circulantGraphs/ (accessed: 21.03.2019).

A. Yu. Romanov, PhD, Associate Professor, e-mail: a.romanov@hse.ru,
E. A. Vedmid, Student, e-mail: eavedmid@edu.hse.com,
National Research University Higher School of Economics, Moscow, Russian Federation,
E. A. Monakhova, PhD, Associate Professor, Senior Researcher, e-mail: emilia@rav.sscc.ru,
ICMMG SB RAS, Novosibirsk, Russian Federation

Designing Networks-on-Chip Based on Triple Loop (Circulant) Networks: Routing Algorithm Development

This paper presents implementation of several dynamic routing algorithms designed for using in networks-on-chip based on circulant topology of type C(N; 1, s_2, s_3) to search for the shortest routes between nodes. The developed algorithms can be implemented as RTL state machine for choosing the direction of packets in routers. Algorithms were tested on various sets of optimal triple loop circulants and compared in terms of efficiency, speed, and resources held in memory. The relationship between efficiency and the difference between the two generatrices was obtained, and the most effective one was found — the coefficient search algorithm. For all tested circulants, algorithm shows maximum efficiency, but the execution time of this algorithm is significantly higher than its considered counterparts. In addition, the efficiency and speed of the algorithm directly depend on the chosen calculation coefficients. Compared with the classic Dijkstra algorithm, the proposed algorithms do not require calculation of the entire packet path, but determine the port number to which the packet should be sent, so that it can reach the destination node. This makes it possible to significantly simplify the structure of the network-on-chip router.

Keywords: *network-on-chip, Dijkstra's algorithm, third loop circulants, routing algorithms*

DOI: 10.17587/it.25.522-530

References

1. **Dahir N. S.** et al. Modeling and tools for power supply variations analysis in networks-on-chip, *IEEE Trans. Comput.*, 2014, vol. 63, no. 3, pp. 679—690.

2. **Benmessaoud Gabis A., Koudil M.** NoC routing protocols — objective-based classification, *J. Syst. Archit. Elsevier B. V.,* 2016, vol. 66—67, pp. 14—32.

3. **Paul J.** et al. Resource-awareness on heterogeneous MPSoCs for image processing, *J. Syst. Archit.*, 2015, vol. 61, no. 10, pp. 668—680.

4. **Abdelfattah M. S., Bitar A., Betz V.** Design and Applications for Embedded Networks-on-Chip on FPGAs, *IEEE Trans. Comput.,* 2017, vol. 66, no. 6, pp. 1008—1021.

5. **Angiolini F.** et al. A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs, *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2007, vol. 26, no. 3, pp. 421—434.

6. **Marvasti M. B., Szymanski T. H.** The performance of hypermesh NoCs in FPGAs, *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 2012, pp. 492—493.

7. **Bishnoi R.** et al. Distributed adaptive routing for spidergon NoC, *18th International Symposium on VLSI Design and Test, VDAT 2014*, 2014, pp. 1—6.

8. **Deb D.** et al. Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines, J. *Parallel Distrib. Comput. Academic Press*, 2019, vol. 123, pp. 118—129.

9. **Dally W. J., Towles B. P.** Principles and Practices of Interconnection Networks. Elsevier, 2003, 581 p.

10. **Kumar A., Tyagi S., Jha C. K.** Performance analysis of network-on-chip topologies, *J. Inf. Optim. Sci.*, 2017, vol. 38, no. 6, pp. 989—997.

11. **Vilfred V.** A few properties of circulant graphs: Self-complementary, isomorphism, Cartesian product and factorization, *2017 7th International Conference on Modeling, Simulation, and Applied Optimization, ICMSAO 2017,* 2017, pp. 1—5.

12. **Monakhova E. A.** Structural and communicative properties of circulant networks, *Prikladnaya Diskretnaya Matematika*, 2011, vol. 3, no. 13, pp. 92—115 (in Russian).

13. **Cai J. Y.** et al. On routing in circulant graphs, *Lecture Notes in Computer Science,* Springer, Berlin, Heidelberg, 1999, vol. 1627, pp. 360—369.

14. **Kuznetsov N. A., Fetisov V. N.** Algoritm Dijkstry s uluchshennoj robastnost'ju dlja upravlenija marshrutizaciej v IP-setjah, *Avtom. i Telemeh.*, 2008, vol. 2, pp. 80—85 (in Russian).

15. **Monakhova E. A.** Multiplicative circulant networks**,** *Diskretniy Analiz i Issledovanie Operaciy*, 2010, vol. 17, pp. 56—66 (in Russian).

16. **Romanov A. Yu., Romanova I. I., Glukhikh A. Yu.** Development of a Universal Adaptive Fast Algorithm for the Synthesis of Circulant Topologies for Networks-on-Chip Implementations, *2018 IEEE 38th International Scientific Conference on Electronics and Nanotechnology, ELNANO 2018*, 2018, pp. 110—115.

17. **Monakhova E. A., Monakhov O. G.** On some characteristics of circulant and toroidal structures of computing systems, *Vestnik SibGUTI*, 2013, no. 3, pp. 63—69 (in Russian).

18. **Dijkstra E. W.** A note on two problems in connexion with graphs, *Numer. Math.,* 1959, vol. 1, pp. 269—271.

19. **Levitin A. V.** Greedy methods: Dijkstra's Algorithm, *Algoritmi. Vvedenie v Razrabotku i Analiz*, 2006, pp. 189—195 (in Russian).

20. **Barrière L.** et al. Fault-tolerant routings in chordal ring networks, *Networks,* 2000, vol. 36, no. 3, pp. 180—190.

21. **Liestman A. L., Opatrny J., Zaragoza M.** Network properties of double and triple fixed step graphs, *Int. J. Found. Comput. Sci.,* 1998, vol. 9, no. 1, pp. 57—76.

22. **Monakhova E. A.** Optimal Triple Loop Networks with Given Transmission Delay: Topological Design and Routing, *International Network Optimization Conference,* Paris, 2003, pp. 410—415.

23. **Romanov A. Yu.** Development of routing algorithms in networks-on-chip based on ring circulant topologies, *Heliyon,* Elsevier, 2019, vol. 5, no. 4, pp. e01516.

24. **Romanov A. Yu.** Optimal Circulants Dataset, available at: https://github.com/RomeoMe5/circulantGraphs/ (accessed: 21.03.2019).