УДК 004.722 DOI 10.31114/2078-7707-2018-3-119-125

Разработка алгоритма маршрутизации в сетях на кристалле с топологией мультипликативный циркулянт

М.А. Щеголева, А.Ю. Романов

Национальный исследовательский университет «Высшая школа экономики», г. Москва, mariyashcheg@gmail.com, a.romanov@hse.ru

***Аннотация* — Разработка многоядерных процессорных систем является востребованным направлением науки и техники. Появление процессоров с десятками и сотнями ядер ставит перед разработчиками вопрос о выборе оптимальной топологии, способной обеспечить эффективную маршрутизацию в сети с большим количеством узлов. В настоящей работе рассматривается возможность применения мультипликативных циркулянтов в качестве топологии для сетей на кристалле. Предлагается способ организации адресного поля пакета при статической маршрутизации на основе стандартного алгоритма поиска кратчайшего пути в сети с циркулянтной топологией. Разработан специализированный алгоритм маршрутизации в сетях с топологией мультипликативный циркулянт, учитывающий особенности топологии и имеющий высокие показатели масштабируемости.**

***Ключевые слова* — сеть на кристалле, топология сети на кристалле, мультипликативный циркулянт, алгоритм маршрутизации.**

I. Введение

В настоящее время одним из важнейших направлений исследований в области информатики и вычислительных систем является построение многоядерных процессоров. Переход к многоядерным процессорам позволяет преодолеть снижение производительности при проектировании все более сложных одноядерных систем [1]. В условиях растущего интереса к технологиям построения систем на кристалле (Systems on Chip, SoCs) и мультипроцессорных систем на кристалле (Multi-Processor Systems on Chip, MPSoCs) приобретают широкое распространение сети на кристалле (Networks-on-Chip, NoCs). В многоядерном процессоре с небольшим количеством ядер (2–8 ядер) коммуникация между IP-ядрами и другими компонентами происходит с помощью общей шины, которая не способна обеспечить коммуникацию между большим количеством ядер: электрическая нагрузка на шину снижает ее рабочую скорость, в результате чего шина уже не отвечает требованиям пропускной способности мультипроцессорных систем на кристалле [1]. Решить проблему масштабируемости позволяет технология построения сетей на кристалле, пришедшая на замену шинным архитектурам.

Одной из актуальных проблем в исследовании сетей на кристалле является поиск оптимальных топологий, поскольку классические регулярные топологии (mesh, torus, hypercube, spidergon) не удовлетворяют современным требованиям к сетям на кристалле, особенно с увеличением количества узлов [2]. Попытка сохранения основных характеристик таких топологий приводит к большим затратам ресурсов.

Циркулянтные топологии имеют лучшие характеристики по сравнению со стандартными топологиями, например, гиперкубами: они обладают лучшими показателями структурной живучести, надежности и связности, а также требуют меньшего числа межпроцессорных обменов при решении вычислительных задач и задач системного управления [3]. Это позволяет использовать их в сетях с большим количеством узлов, насчитывающим десятки и сотни ядер, что уже сейчас с появлением систем с 48, 80 и больше ядрами [4]–[5] является насущной необходимостью. Ряд известных семейств циркулянтов хорошо описан в научной литературе, например, рекурсивные циркулянты [6], а также их подвид – мультипликативные циркулянты [7]. Для применения их в качестве топологии для сетей на кристалле необходимо разработать алгоритмы маршрутизации в них, учитывающие особенности данных семейств графов и организации сетей на кристалле, что определяет актуальность данной работы.

II. Мультипликативные циркулянты

Дадим определение циркулянтного графа. Пусть $s_1, s_2, \dots, s_k, n$ – целые числа, такие что $1 \le s_1 < s_2 < \dots < s_k < n/2$. Граф C с множеством вершин $V = \{0, 1, \dots, n-1\}$ и множеством ребер $E = \{(i,j)\colon |i-j| \equiv s_l \bmod n,\ l = \overline{1,k}\}$ называется циркулянтным [8]–[9], числа $S = (s_1, s_2, \dots, s_k)$ – образующими, числа k и n – размерностью и порядком графа [10].

Циркулянтные графы вида $C(n; 1, s, s^2, \dots, s^{\log_s k-1})$, где $s \geqslant 2$, называются рекурсивными циркулянтами, являющиеся частным случаем кольцевых графов. В работе [6] подробно исследованы свойства рекурсивных циркулянтов, у которых $n = cs^k$. В отдельный класс

мультипликативных циркулянтов [7] выделяют циркулянтные графы вида $C(s^k;\ 1,\ s,\ s^2,\ \dots,\ s^{k-1}) = MC(s,\ k)$, у которых $c = 1$ и $n = s^k$ (рис. 1, 2).

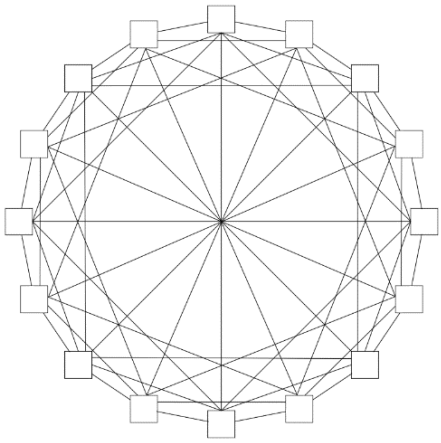

Рис. 1. Мультипликативный циркулянт $MC(2,\ 4)$

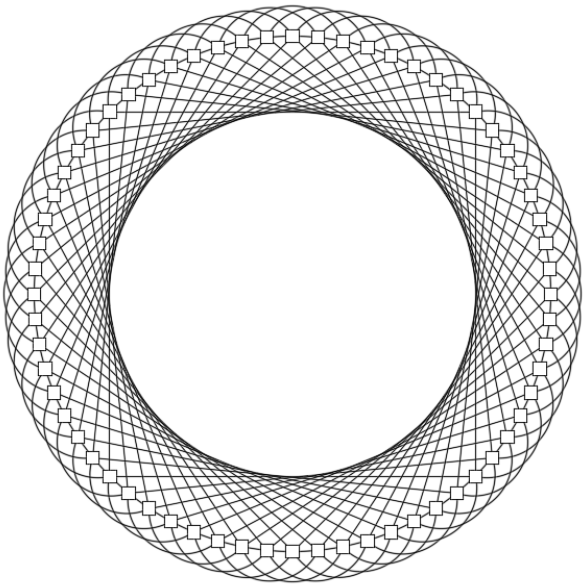

Рис. 2. Мультипликативный циркулянт $MC(4,\ 3)$

Важнейшей характеристикой графа является его диаметр и среднее расстояние. Диаметром графа G называется максимально возможное расстояние (длина кратчайшего пути) между двумя вершинами в этом графе.

$$d(G) = \max_{i,j \in V} d(i,\ j),$$

где $d(i,\ j)$ – длина кратчайшего пути из вершины i в j.

В ряде работ сделаны попытки найти наиболее точные оценки диаметра и среднего расстояния для мультипликативных циркулянтов. В работе [7] приводится формула (1) для вычисления диаметра мультипликативных циркулянтов с четным s:

$$d(MC(s,\ k)) = \frac{ks}{2} - \left\lfloor \frac{k}{2} \right\rfloor. \quad (1)$$

Для циркулянтов с нечетным s проблема нахождения диаметра (2) решена в работе [11]:

$$d\big(MC(s,\ k)\big) = k \left\lfloor \frac{s}{2} \right\rfloor. \quad (2)$$

Также в работах [7], [11] получены оценки среднего расстояния в мультипликативных циркулянтах $MC(s,\ k)$. Для нечетных s значение среднего расстояния можно определить с помощью формулы (3); для четных – по формуле (4):

$$L_{av}\big(MC(s,\ k)\big) = k\frac{s^2-1}{4s} \approx \frac{ks}{4}, \quad (3)$$

$$L_{av}\big(MC(s,\ k)\big) = k\frac{s-1}{4} + \varepsilon,\ \varepsilon \le \frac{k}{4}. \quad (4)$$

Более точные оценки диаметра (5) и среднего расстояния (6) для класса мультипликативных циркулянтов $MC(2,k)$ получены в работе [12]:

$$d\big(MC(2,\ k)\big) = \left\lceil \frac{k}{2} \right\rceil, \quad (5)$$

$$L_{av}\big(MC(2,\ k)\big) \approx \frac{k}{3}. \quad (6)$$

Одной из важнейших характеристик сети, в соответствии с которой оценивается качество выбранной топологии, является ширина бисекции – количество соединений, которые необходимо разорвать, чтобы разделить граф топологии на два эквивалентных непересекающихся подграфа. Данный показатель используется для оценки пропускной способности сети. Определение ширины бисекции для рекурсивных циркулянтных графов рассмотрено в работе [13]; там же получена верхняя оценка ширины бисекции (7).

$$Bw(MC(s,\ k)) \le \sqrt[r]{r^{\log r + \log\log r} * r!} * n^{1-\frac{1}{r}}, \quad (7)$$

где $r = k - 1$.

Сравнение основных характеристик топологий на основе мультипликативных циркулянтов (1–6) и топологии mesh-типа с тем же $n = s^k$ количеством узлов представлено в табл. 1. Для расчета диаметра и среднего расстояния топологии mesh использованы формулы (8–9) [2], [14].

$$d(mesh_n) = 2\big(\sqrt{n} - 1\big), \quad (8)$$

$$L_{av}(mesh_n) = \frac{2(n-1)}{3\sqrt{n}}, \quad (9)$$

Из табл. 1 следует, что даже при небольшом размере сети циркулянтная топология имеет лучшие показатели по всем важнейшим характеристикам сети по сравнению с широко используемой топологией mesh. Таким образом, мультипликативные циркулянты являются хорошим вариантом топологии для проектирования сетей на кристалле, причем с большим количеством ядер, достигающим сотен и даже тысяч. Однако необходимо отметить, что несмотря на хорошую масштабируемость в сторону увеличения размера сети, исходя из особенностей построения, топологии на основе мультипликативных циркулянтов имеют ограниченное число вариантов – количество

узлов должно быть строго степенью целого числа. В противном случае циркулянт становится рекурсивным с другими свойствами и характеристиками [6].

Таблица 1

Сравнение характеристик циркулянтной и mesh топологий

Цирку-лянт $MC(s,k)$	Кол-во узлов n	Диа-метр $d(MC)$	Среднее расстояние $L_{av}(MC)$	Диаметр $d(mesh)$	Средн. расстоя-ние $L_{av}(mesh)$
MC(2,4)	16	2	1,33	6	2,50
MC(2,6)	64	3	2,00	14	5,25
MC(3,4)	81	4	2,67	16	5,93
MC(5,4)	625	8	4,80	48	16,64
MC(3,6)	729	6	4,00	52	17,98
MC(6,4)	1296	10	(5,00, 6,00)	70	23,98
MC(7,4)	2401	12	6,86	96	32,65

III. Разработка структуры пакета при статической маршрутизации в сетях с топологией на основе мультипликативных циркулянтов

В циркулянтных сетях используется парная маршрутизация [10], когда пакет пересылается из узла-источника в узел-приемник. Для организации парной маршрутизации в сетях на кристалле с топологией на основе мультипликативного циркулянта можно воспользоваться стандартными алгоритмами поиска кратчайшего пути, например, широко известным алгоритмом поиска в ширину (breadth-first search, BFS [15]). Предлагается выбрать статический тип маршрутизации, при котором маршрутизатор при каждом узле хранит список смежности. Каждый элемент данного списка соответствует одному из узлов сети и представляет собой еще один список с номерами узлов, с которыми соединен текущий узел. Каждый маршрутизатор знает свой порядковый номер, т.е. номер узла, входящие пакеты которого он распределяет.

На вход маршрутизатору поступает номер узла назначения (узла-приемника). Зная структуру сети маршрутизатор рассчитывает кратчайший путь по алгоритму поиска в ширину. Например, для мультипликативного циркулянта $MC(4,\ 3) = C(64;\ 1,\ 4,\ 16)$ (рис. 2) для движения из узла 5 в узел 17 алгоритм предложит путь $5 \to 21 \to 17$. Далее маршрутизатор перезаписывает путь в обратном порядке и заменяет номера узлов на номера портов, по которым необходимо направлять пакет. Под портом понимается соединение текущего узла с другими. Каждый узел имеет $2k$ портов: по 2 порта каждой длины из S (кроме циркулянтов $MC(2,\ k)$, где на один порт меньше). Так, у каждого узла циркулянта $MC(4,\ 3)$ имеется по два порта длины 1, два порта длины 4, два порта длины 16. По длине порта и направлению движения (пакет в сети может двигаться как влево – в направлении против часовой стрелки, так и вправо – по часовой стрелке) можно однозначно восстановить номер порта. Таким образом, для циркулянта $MC(4,\ 3)$ путь $5 \to 21 \to 17$ преобразуется в $17 \leftarrow 21 \leftarrow 5$, затем в последовательность действий –(4), +(16), а затем в 2|1 или 010|001 в двоичном коде, если установить следующее правило определения номеров портов:

1: –(16) «влево на 16»;
2: –(4) «влево на 4»;
3: –(1) «влево на 1»;
4: +(1) «вправо на 1»;
5: +(4) «вправо на 4»;
6: +(16) «вправо на 16».

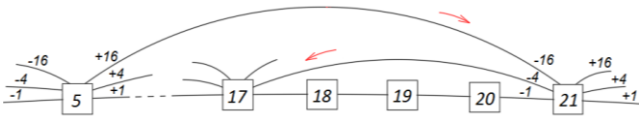

Рис. 3. Путь из узла 5 в узел 17 для циркулянта $MC(4,\ 3)$

Замена номеров узлов на номера портов позволяет уменьшить размер части пакета, отводимой для хранения пути. Каждый узел читает последние k бит, сдвигает путь на это же число бит вправо и передает пакет дальше. Пакет достигает назначения, когда весь путь заполнен нулями. Функции формирования списков смежности и поиска кратчайшего пути по алгоритму BFS, описанные на языке программирования Python, представлены на рис. 4, 5.

```
def form_adjacency_list(s, k):
    n = s**k
    edges = [s**i for i in range(0, k)]
    adj_list = {}
    for i in range(1, nodes + 1):
        temp = []
        # если из вершины идти вперед
        for e in edges:
            if i + e > nodes:
                step = e - (n - i)
            else:
                step = i + e
            temp.append(step)
        temp = sorted(temp, reverse=True)
        # если из вершин идти назад
        temp2 = []
        for e in edges:
            if i - e < 1:
                step = n + (i - e)
            else:
                step = i - e
            temp2.append(step)
        temp2 = sorted(temp2)
        adj_list[i] = list(j for i in zip(temp,
temp2) for j in i)
    return adj_list
```

Рис. 4. Функция формирования графа на языке Python

```
def bfs_paths(G, start, goal):
    queue = [(start, [start])]
    while queue:
        (v, path) = queue.pop(0)
        to_iter = [item for item in G[v] if item
not in path]
        for next in to_iter:
            if next == goal:
                yield path + [next]
            else:
                queue.append((next, path+[next]))


def shortest_path(G, start, goal):
    return next(bfs_paths(G, start, goal))
```

Рис. 5. Функция поиска кратчайшего пути по алгоритму BFS на языке Python

Данный алгоритм универсален и подходит для сетей, построенных на основе любых мультипликативных циркулянтов с различными s и k. Проблема данного алгоритма состоит в том, что с увеличением количества узлов и соединений время работы алгоритма значительно увеличивается. Также увеличивается размер адресной части пакета, поэтому для больших сетей на кристалле требуется разработать специализированный алгоритм, оптимизированный для данного типа топологии.

IV. Разработка специализированного алгоритма маршрутизации для сетей на кристалле на основе топологии мультипликативный циркулянт

Обзор применяемых в сетях на кристалле алгоритмов маршрутизации [16] показывает, что наиболее распространенным в сетях с mesh-подобной топологией является XY алгоритм. Особенность алгоритма заключается в его простоте и детерминированном характере, поскольку в нем учитывается регулярность топологии mesh и заранее известная прямоугольная форма сети, что позволяет осуществлять передачу пакетов сначала в горизонтальном направлении, а потом – в вертикальном. Таким образом, маршрутизаторам нет необходимости хранить таблицы маршрутизации, а достаточно сравнивать номер узла назначения пакета со своим номером, после чего на основе простого алгоритма пакет направляется в нужный порт. В адресной части пакета достаточно хранить только номер узла назначения, тем самым уменьшая объем передачи служебной информации по сети.

Мультипликативные циркулянты тоже обладают строгой, заранее известной геометрической формой. Следовательно, если учесть особенность данной топологии, состоящую в том, что длины образующих являются степенями одного основания, можно предложить специализированный алгоритм маршрутизации, который позволяет упростить структуру адресной части пакета и сократить ее размер. Реализация алгоритма на языке Python представлена на рис. 6–7.

```
def define_step(n, edges, current_node,
target_node):
    flag = 1
    # Пересчет номера назначения
    if target_node > current_node:
        target_node -= current_node
    else:
        target_node += (n - current_node)
    # Выбор направления движения
    if target_node > (n // 2):
        target_node = n - target_node
        flag = -1
    # Подбор шага
    i = len(edges) - 1
    while target_node < edges[i]:
        i -= 1
    if abs(target_node - edges[i]) >
abs(target_node - edges[i + 1]):
        step = flag * edges[i + 1]
    else:
        step = flag * edges[i]
    return step
```

Рис. 6. Функция на языке Python, определяющая следующий шаг движения пакета в сети на кристалле с топологией мультипликативный циркулянт

```
def route_packet(s, k, current_node,
target_node):
    edges = [s ** i for i in range(0, k + 1)]
    n = s ** k
    path = [current_node]
    while target_node != current_node:
        step = define_step(n, edges,
current_node, target_node)
        current_node += step
        if current_node < 0:
            current_node = n + current_node
        elif current_node > n:
            current_node -= n
        path.append(current_node)
    return path
```

Рис. 7. Функция на языке Python для моделирования процесса маршрутизации в сети на кристалле с топологией мультипликативный циркулянт

Получая на вход номер узла, куда необходимо доставить пакет, маршрутизатор текущего узла рассчитывает не полностью весь путь, а только следующий шаг. Это позволяет отказаться от хранения матриц или списков смежности, сильно усложняющихся и требующих много памяти при увеличении количества узлов в сети. Это является важным преимуществом данного алгоритма при организации передачи данных в сетях на кристалле, поскольку память является дорогим ресурсом с точки зрения занимаемой площади на кристалле и энергопотребления. При этом для вычисления следующего шага маршрутизатору достаточно знать собственный номер, номер узла назначения и характеристики циркулянта – s и k.

С учетом того, что циркулянтная топология циклична, алгоритм построен с позиции нулевого узла. Для этого перед началом работы производится пересчет номера узла назначения, исходя из номера текущего узла: если номер текущего узла больше номера узла назначения, то пересчитанный номер узла назначения равен разности двух номеров. Если же номер узла назначения меньше номера текущего узла, то полученная разность вычитается из количества узлов в сети, чтобы учесть переход через нулевой узел. Так, в сети с топологией на основе мультипликативного циркулянта $MC(4,\ 3)$ (рис. 2) для пакета, который необходимо доставить в 17 узел, маршрутизатор 5 узла после пересчета получит новый номер узла назначения – 12, а для этого же пакета маршрутизатор 19 узла – 61. Затем алгоритм определяет, в каком направлении выгоднее начать движение – влево или вправо. Поскольку циркулянт симметричен, то достаточно рассмотреть половину циркулянта, преобразовав номера узлов, превышающие половину количества узлов в циркулянте, в зеркальные (вычесть из количества узлов номер преобразуемого узла) и зафиксировав выбранное направление движения. Например, для того чтобы доставить пакет из 0 узла в 41 в циркулянте $MC(4,\ 3)$, выгоднее двигаться влево; при этом будут совершены такие же шаги, как и при доставке пакета из 0 узла в 23. Шаг определяется по длине образующей, наиболее близкой к номеру узла назначения. Это позволяет учесть случаи, когда выгоднее перешагнуть узел назначения, а затем вернуться назад. Например, при доставке пакета из

узла 0 в узел 12 выгоднее сделать шаг по образующей длины 16 и вернуться назад по образующей длины 4, чем трижды передавать пакет по образующим длины 4. Для этого выбираются образующая с максимальной длиной, не превышающей номер узла назначения, и следующая по длине образующая с большей длиной. Из двух образующих выбирается та, абсолютное значение разности длины которой с номером узла назначения является минимальным. Пакет достигает узла назначения тогда, когда номер текущего узла равен номеру узла, указанному в адресной части пакета. Таким образом, для маршрутизации в адресной части пакета необходимо предусмотреть $\lceil k \cdot \log_2 s \rceil$ бит для хранения номера узла назначения.

В результате, алгоритм вычисляет пути такой же длины, как и алгоритм поиска в ширину. Для примера с передачей пакета из 5 узла в 17, рассмотренного в предыдущем разделе (рис. 3), будет следующая последовательность шагов:

Этап 1. Маршрутизатор 5 узла: пересчитанный номер узла назначения 12, шаг вправо длины 16 (порт 6) → пакет передается в 21 узел.

Этап 2. Маршрутизатор 21 узла: пересчитанный номер узла назначения 60, шаг влево длины 4 (порт 2) → пакет передается в 17 узел. Пакет достиг узла назначения.

Разработанный специализированный для мультипликативных циркулянтов алгоритм требует значительно меньшего времени для вычислений (табл. 2) и легко масштабируется на большие по размерам сети. Данный алгоритм осуществляет выбор шагов с помощью математических операций и имеет линейную сложность в отличие от алгоритма поиска в ширину с экспоненциальной зависимостью времени поиска пути от размеров графа (рис. 8).

Таблица 2

Время, затрачиваемое на поиск кратчайшего пути в мультипликативных циркулянтах

Циркулянт	Кол-во узлов	BFS	Специализированный алгоритм
$MC(2,4)$	16	0,000	0,000
$MC(2,5)$	32	0,002	0,000
$MC(2,6)$	64	0,013	0,001
$MC(3,4)$	81	0,029	0,000
$MC(5,3)$	125	0,340	0,001
$MC(3,5)$	243	2,096	0,004
$MC(6,3)$	216	10,578	0,005
$MC(2,9)$	512	31,037	0,015

В табл. 2 указано суммарное время, затрачиваемое каждым алгоритмом на поиск пути из 0 узла во все остальные. Тестирование проводилось на ноутбуке Acer Aspire V с процессором Intel Core i5–5200U 2.2 ГГц и оперативной памятью 6 Гб.

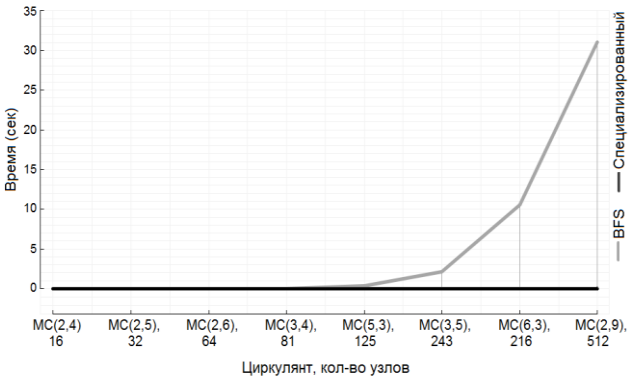

Рис. 8. Время поиска пути специализированным алгоритмом и алгоритмом поиска в ширину

Разработанный алгоритм характеризуется подходом, схожим с алгоритмом маршрутизации, предложенным независимо для рекурсивных циркулянтов в работе [6]. В связи с тем, что полученный в настоящей работе алгоритм разрабатывался исходя из особенностей структуры мультипликативных циркулянтов, являющихся частным случаем рекурсивных циркулянтов, он характеризуется меньшей вычислительной сложностью, но при этом может использоваться только для данного класса циркулянтов.

V. Выводы

В условиях несоответствия характеристик распространенных топологий требованиям современных сетей и ввиду необходимости поиска альтернативных вариантов построения сетей в качестве топологии предлагается рассмотреть особый вид графов – мультипликативные циркулянты. Строгие правила формирования структуры циркулянтов с одной стороны накладывают ограничения на количество узлов в сети, а с другой – делают данные топологии легко масштабируемыми на большие по размерам сети и позволяют улучшить по сравнению с классическими топологиями такие важнейшие характеристики, как диаметр и среднее расстояние в сети.

Для мультипликативных циркулянтов применимы стандартные алгоритмы поиска кратчайшего пути, и в настоящей работе предложен способ организации адресной части пакета для уменьшения его размера при статической маршрутизации. В то же время большинство стандартных алгоритмов поиска кратчайшего пути построены на обходе графа, поэтому с увеличением сложности циркулянта время работы алгоритма резко возрастает. С учетом особенностей строения мультипликативных циркулянтов разработан специализированный для данного класса циркулянтов алгоритм, позволяющий значительно упростить структуру адресной части пакета и уменьшить время поиска оптимального пути в графе.

Поддержка

Исследование осуществлено в рамках Программы фундаментальных исследований НИУ ВШЭ в 2018 году.

Литература

[1] Nychis G., Fallin C., Moscibroda T., Mutlu O. Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need? // Hotnets-IX: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ACM. 2010.

[2] Dally W.J., Towles B. Principles and practices of interconnection networks. Elsevier, 2004. 550 p.

[3] Монахова Э.А. Структурные и коммуникативные свойства циркулянтных сетей // Прикладная дискретная математика. 2011. № 3(13). С. 92–115.

[4] Single-chip cloud computer // Intel Corporation, 2009. URL: https://www.intel.ru/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-single-chip-cloud-overview-paper.pdf (дата обращения 20.03.2018).

[5] Intel's Teraflops Research Chip // Intel Corporation. URL: http://download.intel.com/pressroom/kits/Teraflops/Teraflops_Research_Chip_Overview.pdf (дата обращения 20.03.2018).

[6] Park J.-H., Chwa K.-Y. Recursive Circulant: a New Topology for Multicomputer Networks // Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), Kanazawa, Japan, IEEE Computer Society Press. 1994. P. 73–80.

[7] Stojmenovic I. Multiplicative circulant networks. Topological properties and communication algorithms // Discrete Applied Mathimatics. 1997. N. 77. P. 281–305.

[8] Elspas B., Turner J. Graphs with circulant adjacency matrices // Journal of Combinatorial Theory. 1970. V. 9. Iss. 3. P. 297–307.

[9] Boesch F., Tindell R. Circulants and their connectivities // Journal of Graph Theory. 1984. V. 8. I. 4. P.487–499.

[10] Монахова Э.А. Мультипликативные циркулянтные сети // Дискретный анализ и исследование операций. 2010. Т. 17. № 5. С. 56–66.

[11] Wong C.K., Coppersmith D. A combinatorical problem related to multimodule memory organizations. 1974. P. 392–402.

[12] Arno S., Wheeler F.S. Signed digit representations of minimal Hamming weight // IEEE Transactions on Computers. 1993. V. 42. I. 8. P. 1007–1010.

[13] Mans B., Shparlinski I. Random Walks, Bisecting and Gossiping in Circulant Graphs // Algorithmica. 2014. V. 70. I. 2. P. 301–325.

[14] Suboh S., Bakhouya M., Gaber J., El-Ghazawi T. An interconnection architecture for network-on-chip systems // Telecommunication Systems. 2008. V. 37. I. 1–3. P. 137–144.

[15] Bundy A., Wallen L. Breadth-First Search // Catalogue of Artificial Intelligence Tools. 1984. P. 13.

[16] Gabis A.B., Koudil M. NoC routing protocols – objective-based classification // Journal of Systems Architecture. ISSN 1383-7621. 2016. V. 66–67. P. 14–32.

Development of Routing Algorithm in Networks-on-Chip with a Multiplicative Circulant Topology

M.A. Shchegoleva, A.Yu. Romanov

National Research University Higher School of Economics, Moscow

mariyashcheg@gmail.com, a.romanov@hse.ru

***Abstract* — Nowadays, construction of multi-core processors is becoming one of the most popular areas of investigation in computer science field; transition to multi-core processors allows overcoming the performance decrease, observed in complex single-core system design. Increase in core number, however, raise the issue of choosing the best topology, because classic topologies (mesh, hypercube, torus) fail to meet the requirements of modern networks with numerous cores. In this paper, multiplicative circulants, as a possible topology for networks-on-chip, are considered. Comparison of main characteristics of chosen type of circulants with characteristics of widely used mesh topology, makes it possible to consider multiplicative circulants to be a better topology for multi-core systems and to suggest the packet design for simple static routing technique based on classic breadth first search (BFS) algorithm. However, universal solutions have never been the best for a certain class of objects, so a specialized routing algorithm, taking into account the peculiarities of multiplicative circulants, was elaborated. By utilizing only mathematic operations, the developed algorithm managed to avoid exponential dependency on number of nodes inherent to BFS algorithm and demonstrated good performance even for networks with hundreds of nodes. Moreover, the presented algorithm required less service data in the packet, because only the target node number was needed for proper work.**

***Keywords* — network-on-chip, network-on-chip topology, multiplicative circulant, routing algorithm.**

REFERENCES

[1] Nychis G., Fallin C., Moscibroda T., Mutlu O. Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need? // Hotnets-IX: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ACM. 2010.

[2] Dally W.J., Towles B. Principles and practices of interconnection networks / Elsevier, 2004. 550 p.

[3] Monakhova E.A. Strukturnye i kommunikativnye svojstva cirkulyantnyh setej (Structural and communicative properties of circulant networks) // Prikladnaya diskretnaya matematika. 2011. N. 3(13). P. 92–115.

[4] Single-chip cloud computer // Intel Corporation. 2009. URL: https://www.intel.ru/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-single-chip-cloud-overview-paper.pdf (access date 20.03.2018).

[5] Intel's Teraflops Research Chip // Intel Corporation. URL: http://download.intel.com/pressroom/kits/Teraflops/Teraflops_Research_Chip_Overview.pdf (access date 20.03.2018).

[6] Park J.-H., Chwa K.-Y. Recursive Circulant: a New Topology for Multicomputer Networks // Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), Kanazawa, Japan, IEEE Computer Society Press. 1994. P. 73–80.

[7] Stojmenovic I. Multiplicative circulant networks. Topological properties and communication algorithms // Discrete Applied Mathimatics. 1997. N. 77. P. 281–305.

[8] Elspas B., Turner J. Graphs with circulant adjacency matrices // Journal of Combinatorial Theory. 1970. V. 9. I. 3. P. 297–307.

[9] Boesch F., Tindell R. Circulants and their connectivities // Journal of Graph Theory. 1984. V. 8. I. 4. P.487–499.

[10] Monakhova E.A. Mul'tiplikativnye cirkulyantnye seti (Multiplicative circulant networks) // Diskretnyj analiz i issledovanie operacij. 2010. T. 17. N. 5. P. 56–66.

[11] Wong C.K., Coppersmith D. A combinatorical problem related to multimodule memory organizations. 1974. P. 392–402.

[12] Arno S., Wheeler F.S. Signed digit representations of minimal Hamming weight // IEEE Transactions on Computers. 1993. V. 42. I. 8. P. 1007–1010.

[13] Mans B., Shparlinski I. Random Walks, Bisecting and Gossiping in Circulant Graphs // Algorithmica. 2014. V. 70. I. 2. P. 301–325.

[14] Suboh S., Bakhouya M., Gaber J., El-Ghazawi T. An interconnection architecture for network-on-chip systems // Telecommunication Systems. 2008. V. 37. I. 1–3. P. 137–144.

[15] Bundy A., Wallen L. Breadth-First Search // Catalogue of Artificial Intelligence Tools. 1984. P. 13.

[16] Gabis A.B., Koudil M. NoC routing protocols – objective-based classification // Journal of Systems Architecture. ISSN 1383-7621. 2016. V. 66–67. P. 14–32.