УДК 004.722 DOI: 10.31114/2078-7707-2020-4-23-30

Модификация высокоуровневой модели NoCModel 2.0 для моделирования сетей на кристалле с циркулянтными топологиями

П.М. Прилепко, А.Ю. Романов, Е.В. Лежнев

Национальный исследовательский университет «Высшая школа экономики», г. Москва

pmprilepko@edu.hse.ru, a.romanov@hse.ru, elezhnev@hse.ru

***Аннотация* — Процесс проектирования подсистемы связи сетей на кристалле (СтнК) в общем виде состоит в определении шести базовых характеристик сети: топологии (организация связи между элементами СтнК); маршрутизации (определение путей перемещения данных в сети); переключения (способ передачи данных в сети); управления потоком (выделение каналов передачи данных в сети); буферизации (способ промежуточного хранения пакетов); арбитража (планирование использования каналов и буферов). Эти шесть основных характеристик, помимо других менее важных, создают большое архитектурное пространство, которое определяет огромное количество вариантов организации СтнК. В данном исследовании была предложена и выполнена модификация высокоуровневой модели СтнК NoCModel 2.0 для обеспечения проведения моделирования циркулянтных топологий и были проведены эксперименты, с помощью которых показаны корректность и полезность такой модели для различных применений.**

***Ключевые слова* — сеть на кристалле, топология сети на кристалле, высокоуровневое моделирование, маршрутизация.**

I. Введение

Системы на кристалле (СтнК), а также связанная с этим направлением научная проблематика, находятся в фокусе внимания исследователей из многих стран мира в течение последних 10 лет. Эти вопросы отражены во множестве технических решений и вариантов реализаций многопроцессорных систем. СтнК, пришедшие на замену шинным архитектурам, призваны решить проблему масштабируемости многопроцессорных систем, а также обеспечить достаточную пропускную способность подсистемы связи вместе с уменьшением аппаратурных затрат.

Классические регулярные топологии (mesh [1], torus [2], hypercube [3], spidergon [4]) не удовлетворяют современным требованиям к СтнК, особенно с увеличением количества узлов [5]. Одним из наиболее перспективных классов топологий в контексте задачи нам представляются циркулянтные топологии, которые ранее не применялись в проектировании СтнК, но хорошо себя зарекомендовали в сетях передачи данных.

Циркулянт – это неориентированный граф, состоящий из множества вершин и множества образующих. По ребрам, соответствующим образующим, осуществляется переход из одной вершины в другую, и таким образом формируется маршрут из начальной вершины в конечную (рис. 1).

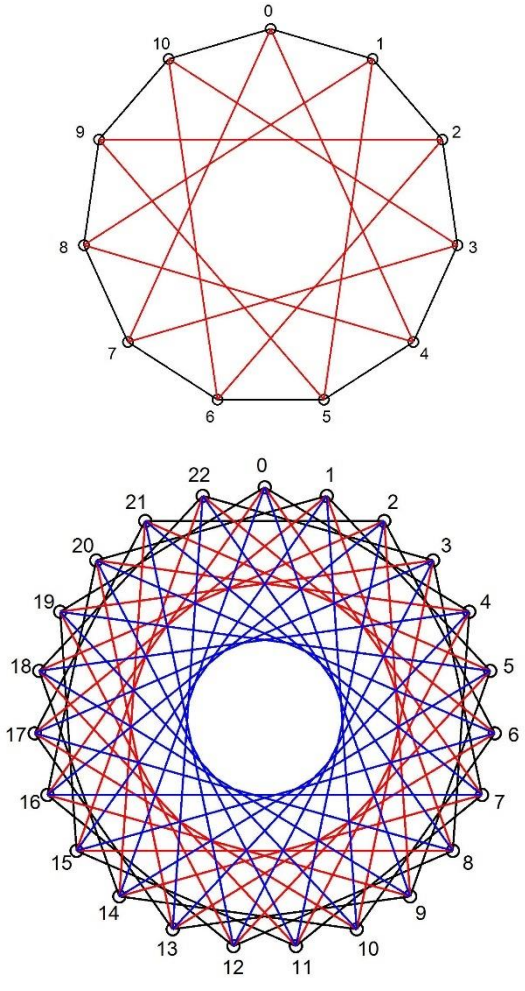

Рис. 1. Циркулянты $C(11; 1, 4)$ и $C(23; 4, 7, 9)$

Циркулянтные топологии имеют лучшие характеристики по сравнению со стандартными топологиями, например, гиперкубами: они обладают

показателями лучшей структурной живучести, надежности и связности, а также требуют меньшего числа межпроцессорных обменов при решении вычислительных задач и задач системного управления [6]. Данные характеристики обусловлены структурой самого циркулянтного графа и его симметричностью. Это хорошо демонстрируется на рис. 2, где показана зависимость диаметра между узлами в хопах от количества узлов в топологии циркулянта $C(N; s_1, s_2)$ в сравнении с классическими регулярными топологиями. При этом циркулянты сохраняют регулярную структуру, что облегчает маршрутизацию в них в сравнении, например, с нерегулярными топологиями [7].

Описанные выше свойства позволяют использовать циркулянты в больших сетях с десятками и сотнями вычислительных узлов, что уже сейчас с появлением систем с 48, 80 и больше ядрами [8] является насущной необходимостью.

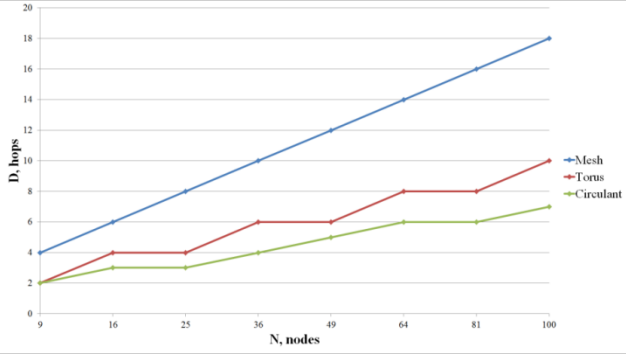

Рис. 2. Зависимость диаметра от количества узлов

Ряд известных семейств циркулянтов хорошо описан в научной литературе, например: рекурсивные циркулянты [9], а также их подвиды – мультипликативные циркулянты [10], кольцевые циркулянты и другие. Для некоторых типов циркулянтов (размерности 2) известны формулы для нахождения оптимальных циркулянтов, то есть циркулянтов с минимальным диаметром (диаметр графа – это наибольшее расстояние между всеми парами вершин графа); для других – необходима разработка программных средств для их поиска. Применительно к топологии сети граф должен быть оптимальным, чтобы маршруты из одной вершины в другую имели меньшую длину. Само понятие «оптимального графа» приводится с точки зрения реализации структуры графа.

Для точных оценок эффективности использования предлагаемых топологий и алгоритмов маршрутизации в таких сетях необходимо разработать новые и модифицировать существующие высокоуровневые модели СтнК, которые бы поддерживали новые топологии.

Для применения циркулянтов в качестве топологий для сетей на кристалле важно разработать алгоритмы маршрутизации в них, учитывающие особенности данных семейств графов и организации сетей на кристалле. В связи с тем, что топология циркулянтов обладает приведенными выше характеристиками, значительно превосходящими графы с другими топологиями, можно предположить, что их использование будет более эффективным, чем остальных, однако для более точной оценки преимуществ таких графов необходимо разработать и провести анализ алгоритмов маршрутизации в таких графах применительно к сетям на кристалле.

Наибольший интерес представляют графы размерности 2 и 3, поскольку применительно к СтнК для их реализации требуются небольшие маршрутизаторы и сравнимое с топологиями mesh и torus количество ресурсов на каналы связи между узлами.

II. Высокоуровневые модели для моделирования сетей на кристалле с циркулянтными топологиями

Моделирование является одним из основных этапов в разработке СтнК. Целью моделирования является оценка основных характеристик сети: оценка расхода ресурсов кристалла, анализ пропускной способности, энергопотребления, временных задержек сети и др. Можно выделить несколько способов моделирования, отличающихся по уровню абстракции:

1) высокоуровневое – моделирование работы сети с использованием языков высокого уровня [11], [12];

2) низкоуровневое – исследование использования ресурсов кристалла и энергопотребления проектируемой системы на уровне ее прототипа, описанного на HDL (Hardware Description Language, язык описания аппаратуры) [13], [14];

3) среднеуровневое – моделирование сетевого трафика и анализ пропускной способности, задержек передачи пакетов, анализ эффективности алгоритмов маршрутизации в СтнК [15] – [17]. К нему также можно отнести гибридные модели, объединяющие несколько уровней абстракции.

В следующем разделе рассматриваются высокоуровневые модели сетей на кристалле.

III. Высокоуровневые модели сетей на кристалле

При моделировании СтнК важной задачей является выбор подходящего симулятора. Разработано множество симуляторов, на основе которых можно проводить исследования параметров СтнК. Их можно разделить на две категории:

- Симуляторы сетей общего назначения – предназначены для моделирования любых сетей в целом, но могут быть использованы и для СтнК. К ним относятся: Gem5, Graphite, Sniper, Hornet, Fusionsim, Esesc, Wattch, Hotspot, NS2, NS3, Omnet ++, Netsim, Orion [18], [19].
- Симуляторы, разработанные специально для моделирования СтнК: BookSim, WormSim, Vnoc, Matrics, SICOSYS, Garnet, Ocintsim, Noxim, Nostrum,

Nirgam, Occn, Nocsim, Access Noxim, NoCTweak, Atlas, gpNoCsim, Xmulator, Phoenixsim, SUNMAP, NOCMAP, ReliableNoC, MapoNoC [11], [12], [20], [21].

В общем виде структура симуляторов СтнК изображена на рис. 3 и может содержать в себе следующие блоки:

1) настройка симулятора (инициализация модели, установка ограничений на время симуляции и время разогрева, выбор режима симуляции);

2) настройка параметров синтетических тестов (размера и структуры сети, профиля трафика, расположения горячих точек сети и т.д.);

3) выбор параметров встроенных тестов (определение характеристического графа задачи и алгоритма проекции графа задачи на ядро);

4) настройка трафика (количества потоков, длины пакета, скорости передачи пакетов);

5) настройка маршрутизатора (количества и длины буферов пакетов, вида маршрутизаторов);

6) настройка маршрутизации (алгоритма маршрутизации);

7) выбор измеряемых параметров (пропускной способности сети, потребления энергии, задержек прохождения пакетов в сети).

Рис. 3. Общая структура симуляторов сетей на кристалле

IV. Высокоуровневая модель СтнК NoCModel 2.0

Среди приведенных выше моделей СтнК NoCModel 2.0 [22], [23] предоставляет ряд перспективных возможностей для моделирования различных 2D-топологий благодаря простоте своей структуры. Вследствие того, что симулятор выводит подробную информацию о процессе симуляции, возможна дальнейшая оценка различных параметров сети. Так как модель распространяется с открытым исходным кодом, она может быть основой для дальнейшей модификации.

NocModel 2.0 представляет собой библиотеку функций для симуляции различных топологий. Библиотека разбита на отдельные модули, что обеспечивает возможности модификации и дальнейшего расширения функционала. Структура модулей симулятора представлена на рис. 4.

Библиотека разделена на 7 частей:

- классы для создания базовых объектов СтнК;
- функции для графического отображения топологии;
- модуль симуляции Transaction Based Model;
- вспомогательные функции к предыдущему модулю;
- модуль базовой поддержки генерации кода;
- модуль VHDL-поддержки генерации кода;
- модуль вспомогательных функций и пакет, состоящий из 5 дополнительных модулей, каждый из которых содержит базовый пример класса объектов СтнК.

Каждая часть библиотеки представляет собой отдельный файл, содержащий классы и/или функции, который может импортироваться при работе программы, а пакет – каталог модулей и/или пакетов.

Модуль noc_base содержит класс-контейнер NoC, основанный на граф-объекте и представляющий собой описание структуры СтнК. Класс-контейнер содержит классы, описывающие базу для объектов СтнК:

- класс ядра, содержащий данные о его положении и наличии связей с роутером;
- класс роутеров, содержащий индекс роутера, точно определяющий его местоположение, и данные о наличии связей с другими роутерами;
- класс каналов, содержащий данные о связи канала и двух роутеров или же ядра и роутера;
- класс протокола, содержащий методы генерации, кодирования, декодирования и интерпретации пакетов;
- класс пакетов, объекты которого содержат данные пакета.

Модуль noc_guilib содержит функции для графического отображения симулируемой топологии.

Модуль noc_tbm_base содержит одноименный класс, определяющий методы отправки и получения пакетов ядром, класс noc_tbm_simulation, управляющий моделированием MyHDL для объекта класса NoC и обеспечивающий поддержку его протоколирования, а также класс noc_tbm_errcodes, определяющий коды основных ошибок.

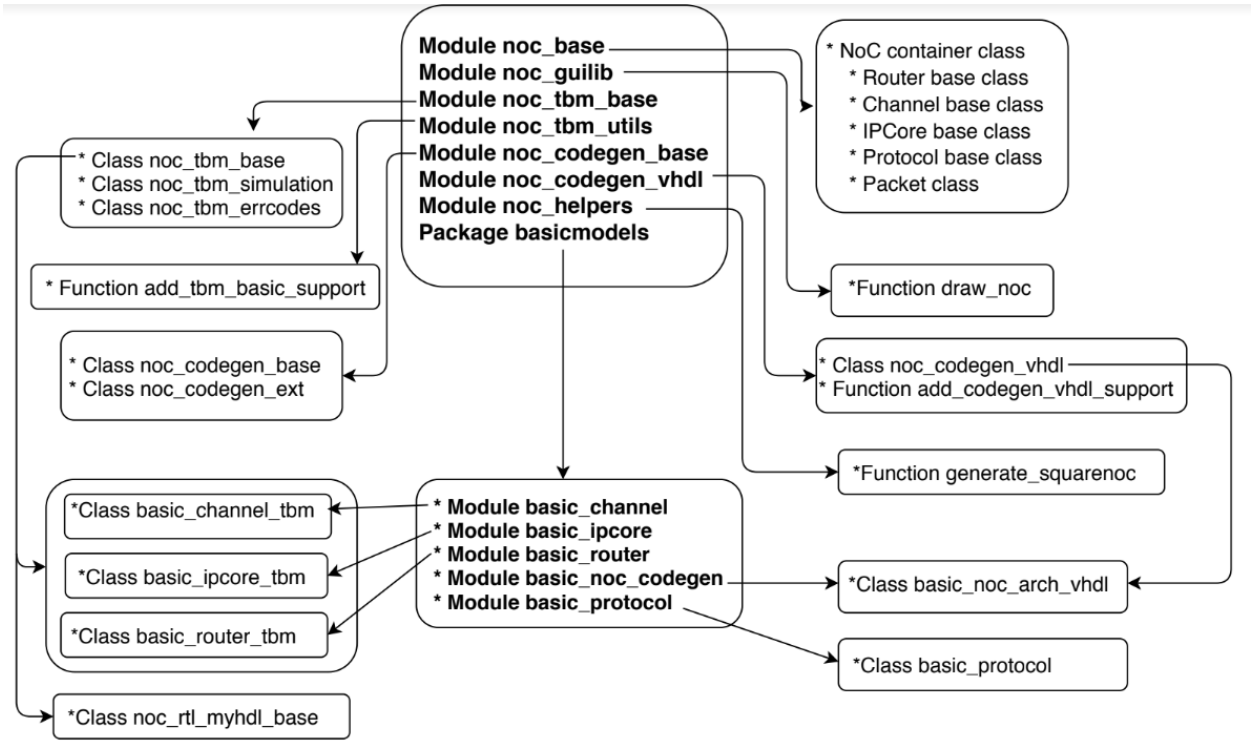

Рис. 4. Структура симулятора NocModel 2.0

Модуль noc_tlm_utils содержит вспомогательную функцию для предыдущего модуля, добавляющую в объект класса NoC объект класса noc_tbm_base.

Модуль noc_codegen_base содержит одноименный класс, содержащий базовую модель аппаратных средств и расширенный VHDL-генератором кода, и класс noc_codegen_ext, который содержит методы доступа для изменения модели кода, генерируемой классом noc_codegen_base.

Модуль noc_codegen_vhdl содержит одноименный класс, определяющий методы для генерации VHDL кода для любого объекта класса NoC.

Модуль noc_codegen_vhdl содержит одноименный класс, определяющий методы для генерации VHDL-кода для любого объекта класса NoC. noc_helpers содержит вспомогательную функцию, генерирующую граф, на основе которого создается 2D-mesh модель.

Пакет basicmodels состоит из 5 модулей, 3 из которых содержат одноименные классы, наследуемые от класса noc_tbm_base, в которых формируются TBM модели объектов класса NoC: ядра, роутера и канала. В модуле basic_noc_codegen содержится класс basic_noc_arch_vhdl, наследующий от класса noc_codegen_vhdl, который является расширением для генерации кода для объектов класса NoC в VHDL. Объект этого класса создает необходимую архитектуру для соединения всех объектов в один головной VHDL файл.

Модуль basic_protocol состоит из одноименного класса, который упрощает протокол создания объекта (в нем определяются поля пакета).

Настройка симулятора производится с помощью стандартного файла на python, в который импортируется библиотека функций и определяются следующие параметры:

1) Топология: возможно вызвать стандартную функцию создания 2D-mesh топологии и задать ее размеры или же прописать координаты для каждого отдельного маршрутизатора и порядок их соединения.

2) Данные для тестов: возможно прописать количество отправляемых пакетов и передаваемые в них значения или же вызвать стандартную функцию генерации пакетов, которая создаст равное количество векторов для каждого ядра.

Также в файле содержатся реализации двух функций: do_checkvect, которая генерирует векторы для проведения тестов, и sourcechkgen, которая определяет количество потерянных пакетов.

Процесс симуляции состоит из трех этапов: разогрев (стадия, когда статистика не собирается), передача пакетов (сбор статистики работы сети) и анализ. Продолжительность этих этапов зависит от заданной топологии СтнК и количества сгенерированных пакетов. На стадии разогрева происходит настройка сети, после чего наступает фаза передачи пакетов. Все сообщения о потерях пакетов и переполнении очереди выводятся в консоль во время симуляции. В третьей фазе происходит подсчет всех

потерянных пакетов и формируется текстовый файл с подробным описанием хода симуляции, а также списком возникших ошибок и количеством потерянных пакетов для каждого ядра. По окончании симуляции в отдельное окно выводится графическое отображение топологии.

V. Модификация NoCModel 2.0

Перед тем как модифицировать симулятор для моделирования циркулянтных топологий, потребовалось внести ряд изменений, чтобы унифицировать модель для запуска с помощью одного python-файла. В первую очередь была добавлена функция рандомизации для таблицы маршрутизации в модуль noc_base в класс router: ранее пути от одного роутера до другого сортировались определенным образом, и по этой причине, даже при наличии альтернативных путей такой же длины, всегда использовались пути, стоящие на первом месте в списке после сортировки, что приводило к переполнению очереди и потере пакетов. После добавления функции рандомизации путь из списка кратчайших путей выбирается случайным образом, из-за чего потери пакетов снизились в 1,5 – 2 раза.

Также определенные изменения были внесены в функцию do_checkvect. Сгенерированные векторы записывались в отдельный файл формата .txt, что, с одной стороны, позволяло при последующих запусках не генерировать векторы повторно, а считывать их из этого файла, что значительно сокращало время симуляции, но с другой – приводило к большому количеству ошибок, так как в файле не указывалось количество узлов топологии, для которого были сгенерированы эти векторы. По этой причине были внесены определенные изменения в формирование названия файла с векторами и функцию считывания векторов из файла: при запуске симуляции после указания количества узлов топологии происходит поиск файла с векторами для этого количества узлов; если файл не найден, то происходит генерация векторов и создание нового файла.

Для моделирования циркулянтных топологий добавлена новая функция в модуль noc_helpers. В качестве аргументов в функцию передается количество узлов топологии и список образующих. Сначала создается граф с количеством вершин, равным заданному количеству узлов, которые расположены по окружности и соединяются в определенном порядке согласно списку образующих. Далее с помощью цикла на место каждого узла помещается маршрутизатор и к нему прикрепляется ядро. На последнем этапе на место ребер графа помещаются каналы, которые соединяют маршрутизаторы между собой.

Для получения более подробных результатов тестирования потребовалось добавить функцию для вывода и сохранения данных в отдельный файл. После каждой симуляции создается новый файл формата .txt или же открывается существующий при его наличии, в который записывается словарь: ключом является количество узлов топологии, а элементом – список из кортежей, содержащих скорость отправки пакетов и скорость приема. Данные о различных топологиях и с различным количеством образующих записываются в разные файлы. Такая система вывода данных позволяет сравнивать различные топологии между собой, а также сохранять данные обо всех запусках каждой топологии, что обеспечивает б**о**льшую достоверность данных.

Для анализа полученных результатов симуляции был создан отдельный python-файл, содержащий функции считывания данных из файлов, а также их обработки: на основе данных о нескольких запусках модели рассчитывается средняя скорость отправки и приема пакетов для каждой из топологий. На последнем этапе строится трехмерный график зависимости скорости отправки и скорости приема пакетов от количества узлов топологии.

VI. Апробация модифицированной модели NoCModel 2.0

Для тестирования NoCModel 2.0 было проведено моделирование сетей с топологиями mesh и циркулянт с количеством образующих 2 и 3. Для данных топологий была выбрана оптимальная форма, чтобы устранить влияние геометрической формы на результат: для тестирования были отобраны квадратные 2D-mesh топологии и циркулянты и датасеты [24], у которых минимальны среднее расстояние между узлами и диаметр. Таким образом было проведено моделирование топологий с 9, 16, 25, 36, 49, 64, 81, 100 и 121 узлами.

Параметры моделирования:

- время настройки сети – 20% от общего времени моделирования;
- максимальное время симуляции – 10000 единиц модельного времени;
- виртуальные каналы – нет;
- время от начала симуляции, в течение которого происходит отправка пакетов, – 900 единиц модельного времени;
- пауза между отправками пакетов для каждого ядра – 30 единиц модельного времени.

Последние два параметра определяют количество пакетов, которое будет отправлено из каждого ядра ($N = \lfloor \text{время}/\text{пауза} \rfloor$).

Результаты моделирования приведены на рис. 5 – 7. Они демонстрируют, что скорость приема пакетов у циркулянтов оставалась выше скорости приема пакетов у mesh при сравнении этих топологий с одинаковым количеством узлов (рис. 6). Также скорость отправки пакетов у циркулянтных топологий больше по сравнению с mesh, так как время работы у циркулянтов меньше (рис. 7). Полученные данные доказывают эффективность циркулянтов в качестве топологий для сетей на кристалле в сравнении с mesh.

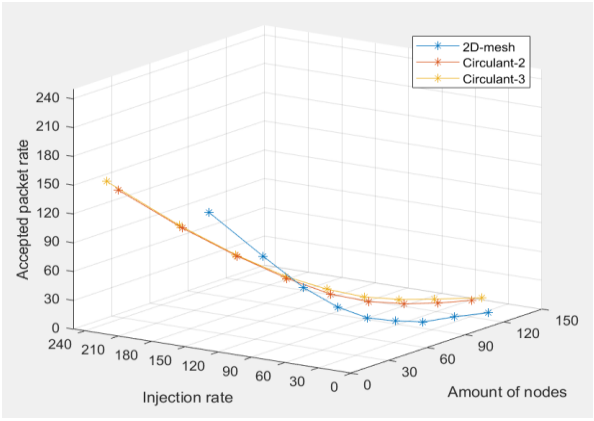

Рис. 5. График зависимости скорости отправки и скорости приема пакетов от количества узлов топологии (общий вид)

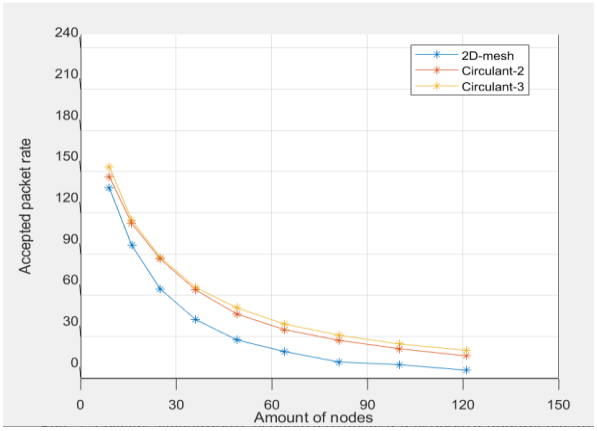

Рис. 6. График зависимости скорости отправки и скорости приема пакетов от количества узлов топологии (вид 1)

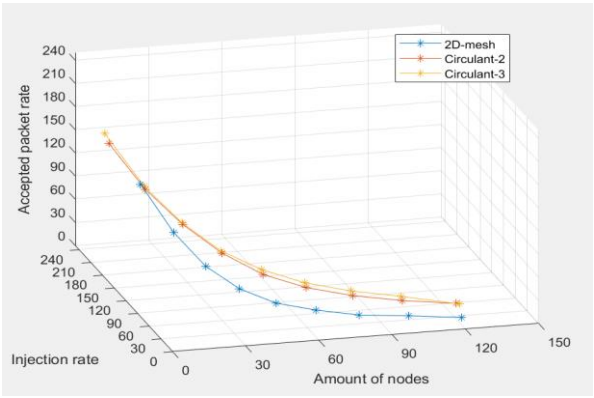

Рис. 7. График зависимости скорости отправки и скорости приема пакетов от количества узлов топологии (вид 2)

VII. Выводы

Модель NocModel 2.0 обладает простой структурой, предоставляющей ряд возможностей для ее модификации. Наличие открытого исходного кода, а также способность проводить моделирование любых топологий СтнК стали еще одной причиной выбора данного симулятора. В процессе модификации в структуру симулятора были внесены изменения, выразившиеся в добавлении новой функции в один из модулей программы для моделирования циркулянтных топологий любого порядка и усовершенствование алгоритма маршрутизации. Также был создан отдельный модуль функций для обработки полученных в процессе тестирования результатов. Для оценки качества работы симулятора было проведено моделирование СтнК с топологиями 2D-mesh и циркулянтов второго и третьего порядка, в результате чего были получены данные по пропускной способности сети, которые соответствуют результатам, полученным с помощью других моделей.

Поддержка

Исследование осуществлено в рамках Программы фундаментальных исследований НИУ ВШЭ в 2020 году.

Литература

[1] Deb D., et al. Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines // J. Parallel Distrib. Comput. 2019. V. 123. P. 118–129.
[2] Ansari A.Q., Ansari M.R., Khan M.A. Modified quadrant-based routing algorithm for 3D Torus Network-on-Chip architecture // Perspect. Sci. 2016. V. 8. P. 718–721.
[3] 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.
[4] Bishnoi R., et al. Distributed adaptive routing for spidergon NoC // 18th International Symposium on VLSI Design and Test, VDAT 2014. 2014. P. 1–6.
[5] Dally W.J., Towles B.P. Principles and Practices of Interconnection Networks. 2004. Elsevier. 581 p.
[6] Monakhova E.A. A Survey on Undirected Circulant Graphs // Discret. Math. Algorithms Appl. World Sci. Publ. Co. 2012. V. 4. № 1. P. 1250002.
[7] Romanov A.Y., Romanova I.I. Use of irregular topologies for the synthesis of networks-on-chip // 2015 IEEE 35th International Conference on Electronics and Nanotechnology (ELNANO). IEEE. P. 445–449.
[8] Comprés Ureña I.A., Riepen M., Konow M. RCKMPI – Lightweight MPI implementation for intel's single-chip cloud computer (SCC) // European MPI Users' Group Meeting. Springer, Berlin, Heidelberg. 2011. P. 208–217.
[9] Park J.-H., Chwa K.-Y. Recursive Circulant: A New Topology for Multicomputer Networks // International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 1994). 1994. P. 73–80.
[10] Stojmenović I. Multiplicative circulant networks topological properties and communication algorithms // Discret. Appl. Math. 1997. V. 77. № 3. P. 281–305.
[11] Jiang N., et al. BookSim 2.0 User's Guide. 11 p.
[12] Tran A., Baas B. Noctweak: A highly parameterizable simulator for early exploration of performance and energy of networks on-chip. 12 p.
[13] Mota R.G., et al. Efficient routing table minimization for fault-tolerant irregular Network-on-Chip // 2016 IEEE International Conference on Electronics, Circuits and Systems, ICECS 2016. 2016. P. 632–635.
[14] Duan X., Li Y. A multiphase routing scheme in irregular mesh-based NoCs // Proceedings – 2011 4th International

Symposium on Parallel Architectures, Algorithms and Programming. 2011. P. 277–280.
[15] Bertozzi D. and other. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip // IEEE Trans. Parallel Distrib. Syst. 2005. V. 16. № 2. P. 113–129.
[16] Goossens K., Dielissen J., Rădulescu A. Æthereal network on chip: Concepts, architectures, and implementations // IEEE Des. Test Comput. 2005. V. 22. № 5. P. 414–421.
[17] Genko N. and other. Feature – NoC emulation: a tool and design flow for MPSoC // IEEE Circuits Syst. Mag. 2007. V. 7. № 4. P. 42–51.
[18] Ben-Itzhak Y., et al. NoCs simulation framework for OMNeT++ // NOCS 2011: The 5th ACM/IEEE International Symposium on Networks-on-Chip. 2011. P. 265–266.
[19] Wang H.S., et al. Orion: A power-performance simulator for interconnection networks // Proceedings of the Annual International Symposium on Microarchitecture, MICRO. 2002. P. 294–305.
[20] Catania V., et al. Cycle-accurate network on chip simulation with Noxim // ACM Trans. Model. Comput. Simul. 2016. V. 27. № 1. Art. No. 4.
[21] Romanov A.Y., Ivannikov A.D., Romanova I.I. Simulation and synthesis of networks-on-chip by using NoCSimp HDL library // 2016 IEEE 36th International Conference on Electronics and Nanotechnology. 2016. IEEE. P. 300–303.
[22] Díaz O. and other. NoCmodel: An extensible framework for Network-on-Chips modeling // 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip, ReCoSoC 2011 – Proceedings. 2011. P. 1–6.
[23] Oscar Diaz. dargor0/nocmodel: Python module for Network-on-Chip modeling, with add-ons for simulation (functional or RTL) and code generation. URL: https://github.com/dargor0/nocmodel (дата обращения: 28.03.2020).
[24] Romanov A.Y., Glukhikh A.Y. circulantGraphs dataset. URL: https://github.com/RomeoMe5/ circulantGraphs (дата обращения: 28.03.2020).

Modification of a High-Level NoCModel 2.0 for Modeling Networks-on-Chip with Circulant Topologies

P.M. Prilepko, A.Yu. Romanov, E.V. Lezhnev

National Research University Higher School of Economics, Moscow

pmprilepko@edu.hse.ru, a.romanov@hse.ru, elezhnev@hse.ru

***Abstract* — The process of designing a communication subsystem of networks-on-chip (NoCs) in its general form is determination of six basic characteristics of a network: topology (organization of communication between NoC elements), routing (ways to move data on the network), switching (method of transmitting data on the network), data flow control (allocation of data transmission channels in the network), buffering (method for intermediate storage of packets), arbitration (planning the use of channels and buffers). These six main characteristics among other less important ones create a large architectural space that defines a huge number of options for NoC organization. Classic regular topologies such as mesh, torus, and hypercube have characteristics that do not meet modern NoC requirements. Thus it is proposed to use circulant topologies for NoC construction. In order to prove the effectiveness of the topological solutions proposed it is necessary to develop high-level models with high speed and sufficient accuracy for NoC work simulation.**

In this study, due to simplicity of its structure, and to ensure the simulation of circulant topologies, a high-level NoCModel 2.0 modification has been proposed. By virtue of its modular structure the selected model provides the possibility of further modification and expansion of its capabilities. New functions for modeling circulant topologies were introduced, a routing algorithm was implemented, and a module for processing results was added. Experiments with optimal shapes of mesh and with circulant topologies having the number of generators 2 and 3 were conducted, and the correctness and usefulness of such a model for various applications was shown. Also the results of topologies simulation showed increasing the speed of receiving and transmitting packets when using circulant topologies instead of mesh with the same number of nodes.

***Keywords* — network-on-chip, network-on-chip topology, high-level simulation, routing.**

References

[1] Deb D., et al. Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines // J. Parallel Distrib. Comput. 2019. V. 123. P. 118–129.
[2] Ansari A.Q., Ansari M.R., Khan M.A. Modified quadrant-based routing algorithm for 3D Torus Network-on-Chip architecture // Perspect. Sci. 2016. V. 8. P. 718–721.
[3] 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.
[4] Bishnoi R., et al. Distributed adaptive routing for spidergon NoC // 18th International Symposium on VLSI Design and Test, VDAT 2014. 2014. P. 1–6.
[5] Dally W.J., Towles B.P. Principles and Practices of Interconnection Networks. 2004. Elsevier. 581 p.
[6] Monakhova E.A. A Survey on Undirected Circulant Graphs // Discret. Math. Algorithms Appl. World Sci. Publ. Co. 2012. V. 4. № 1. P. 1250002.
[7] Romanov A.Y., Romanova I.I. Use of irregular topologies for the synthesis of networks-on-chip // 2015 IEEE 35th International Conference on Electronics and Nanotechnology (ELNANO). IEEE. P. 445–449.
[8] Comprés Ureña I.A., Riepen M., Konow M. RCKMPI – Lightweight MPI implementation for intel's single-chip cloud computer (SCC) // European MPI Users' Group Meeting. Springer, Berlin, Heidelberg. 2011. P. 208–217.

[9] Park J.-H., Chwa K.-Y. Recursive Circulant: A New Topology for Multicomputer Networks // International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 1994). 1994. P. 73–80.
[10] Stojmenović I. Multiplicative circulant networks topological properties and communication algorithms // Discret. Appl. Math. 1997. V. 77. № 3. P. 281–305.
[11] Jiang N., et al. BookSim 2.0 User's Guide. 11 p.
[12] Tran A., Baas B. Noctweak: A highly parameterizable simulator for early exploration of performance and energy of networks on-chip. 12 p.
[13] Mota R.G., et al. Efficient routing table minimization for fault-tolerant irregular Network-on-Chip // 2016 IEEE International Conference on Electronics, Circuits and Systems, ICECS 2016. 2016. P. 632–635.
[14] Duan X., Li Y. A multiphase routing scheme in irregular mesh-based NoCs // Proceedings – 2011 4th International Symposium on Parallel Architectures, Algorithms and Programming. 2011. P. 277–280.
[15] Bertozzi D. and other. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip // IEEE Trans. Parallel Distrib. Syst. 2005. V. 16. № 2. P. 113–129.
[16] Goossens K., Dielissen J., Rădulescu A. Æthereal network on chip: Concepts, architectures, and implementations // IEEE Des. Test Comput. 2005. V. 22. № 5. P. 414–421.
[17] Genko N. and other. Feature – NoC emulation: a tool and design flow for MPSoC // IEEE Circuits Syst. Mag. 2007. V. 7. № 4. P. 42–51.
[18] Ben-Itzhak Y., et al. NoCs simulation framework for OMNeT++ // NOCS 2011: The 5th ACM/IEEE International Symposium on Networks-on-Chip. 2011. P. 265–266.
[19] Wang H.S., et al. Orion: A power-performance simulator for interconnection networks // Proceedings of the Annual International Symposium on Microarchitecture, MICRO. 2002. P. 294–305.
[20] Catania V., et al. Cycle-accurate network on chip simulation with Noxim // ACM Trans. Model. Comput. Simul. 2016. V. 27. № 1. Art. No. 4.
[21] Romanov A.Y., Ivannikov A.D., Romanova I.I. Simulation and synthesis of networks-on-chip by using NoCSimp HDL library // 2016 IEEE 36th International Conference on Electronics and Nanotechnology. 2016. IEEE. P. 300–303.
[22] Díaz O. and other. NoCmodel: An extensible framework for Network-on-Chips modeling // 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip, ReCoSoC 2011 – Proceedings. 2011. P. 1–6.
[23] Oscar Diaz. dargor0/nocmodel: Python module for Network-on-Chip modeling, with add-ons for simulation (functional or RTL) and code generation. URL: https://github.com/dargor0/nocmodel (access date: 28.03.2020).
[24] Romanov A.Y., Glukhikh A.Y. circulantGraphs dataset. URL: https://github.com/RomeoMe5/ circulantGraphs (access date: 28.03.2020).