Аналитический подход к оценке задержек в Mesh-сети в вычислительной системе с распределенной памятью

Д. И. Хайдуков, А. А. Алексеев

***Аннотация*—В работе предлагается аналитическая модель оценки задержек для Mesh-сети вычислительной системы с распределенной памятью, в которой маршрутизаторы моделируются как M/M/1/N. Модель учитывает вероятности коллизий при маршрутизации, при коммутации виртуальных каналов (ВК) и при пакетной передаче по принципу wormhole, а также вероятности переполнения внутренних очередей маршрутизаторов. Она позволяет сопоставлять варианты маршрутизации, число ВК, глубину буферов и размер пакета, направляя проектирование на ранних этапах. Точность подтверждена сопоставлением с потактовым симулятором: ошибка не превышает 5% до насыщения при ускорении порядка 100 раз. Результаты интерпретируемы благодаря разложению общей задержки на вклады входных и выходных буферов маршрутизаторов, а также на компоненты, связанные с трафиком запросов и трафиком ответов. Применение модели к сравнению алгоритмов маршрутизации, размера пакета и параметров емкости сети показало: (i) алгоритм ADOR значительно эффективнее DOR за счет самого алгоритма маршрутизации, а не из-за увеличения емкости (подтверждено при равных аппаратных ресурсах); (ii) пакетная передача эффективнее посылочной, при этом существует оптимальный размер пакета (в проведенных экспериментах — две посылки); (iii) при наращивании емкости более результативно увеличивать число ВК, а не глубину буферов.**

***Ключевые слова*—аналитическая модель, вычислительная система, распределенная память, сеть на кристалле, mesh.**

I. Введение

Современные задачи в таких областях, как компьютерное зрение, цифровая обработка сигналов и искусственный интеллект, предъявляют повышенные требования к производительности и масштабируемости вычислительных систем. Для их эффективного решения требуется параллельная обработка больших объемов данных, что делает востребованными специализированные вычислительные архитектуры. Все более широкое распространение получают гетерогенные системы [1], которые сочетают различные исполнительные устройства и аппаратные ускорители, оптимизированные для выполнения конкретных классов вычислений.

Ключевым элементом таких систем является распределенная память [2]. Она определяет, насколько эффективно исполнительные устройства смогут обмениваться данными и использовать вычислительные ресурсы. С учетом того, что в современных кристаллах доля площади, занимаемая памятью, может превышать 90% [3], особую актуальность приобретает задача построения масштабируемых механизмов связи между вычислительными блоками и памятью [4].

Для организации высокопроизводительного и универсального доступа к памяти внутри кристалла применяются сети на кристалле. Они обеспечивают обмен данными между исполнительными устройствами и модулями памяти, выступая как внутренняя коммуникационная инфраструктура системы. Среди множества вариантов топологий наибольшее распространение получила 2D Mesh [5] благодаря своей регулярной структуре и хорошей масштабируемости (рис. 1). Эффективность такой сети напрямую определяется ее архитектурой и характеристиками маршрутизаторов.

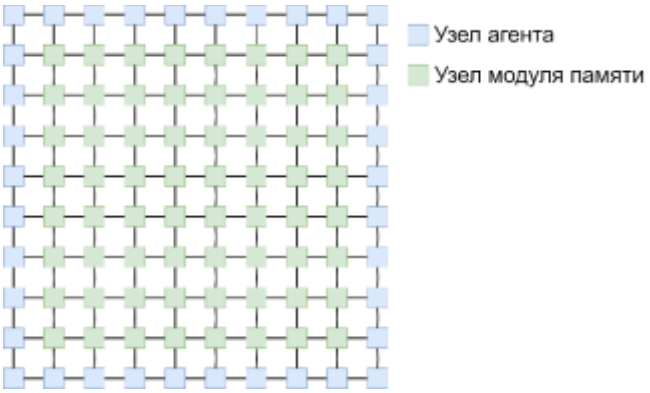


Рис. 1. Mesh-сеть 10x10 вычислительной системы с распределенной памятью

Одним из наиболее критичных показателей для Mesh-сетей является задержка передачи пакетов [6], во многом определяющая производительность вычислительной системы в целом. Важно подчеркнуть, что архитектура сети должна задаваться еще на ранних стадиях

Статья получена 9 октября 2025. Д. И. Хайдуков, Московский Физико-Технический Институт, Москва, Россия (e-mail: khajdukov.di@phystech.edu)

А. А. Алексеев, Московский Физико-Технический Институт, Москва, Россия (e-mail: aleksandr.a.alekseev@phystech.edu)

проектирования, поэтому необходимы методы, позволяющие оценивать задержки при различных вариантах организации сети. В таких случаях активно используются аналитические модели [7], так как они обеспечивают быстрый и достаточно точный прогноз средней задержки без необходимости длительного моделирования на потактовых симуляторах.

В данной работе предложена аналитическая модель для оценки задержек в сетях на кристалле с распределенной памятью. В основе модели лежит представление маршрутизаторов сети как систем массового обслуживания типа M/M/1/N [8]. Такой подход позволяет получать численные оценки средней задержки при различных параметрах архитектуры и характера трафика. В работе впервые предложена аналитическая модель, специально ориентированная на вычислительные системы с распределенной памятью, что составляет научную новизну настоящей работы.

Дальнейшая структура статьи организована следующим образом. В разделе II представлен обзор литературы, включающий анализ архитектурных характеристик сетей на кристалле и существующих аналитических моделей. В разделе III подробно описана предлагаемая аналитическая модель. Алгоритм расчета модели представлен в разделе IV. В разделе V приведено описание исследуемого сценария, а в разделе VI представлены численные результаты проведенных экспериментов. Наконец, в разделе VII сформулированы основные выводы данной работы.

II. Обзор литературы

A. *Архитектурные характеристики сети*

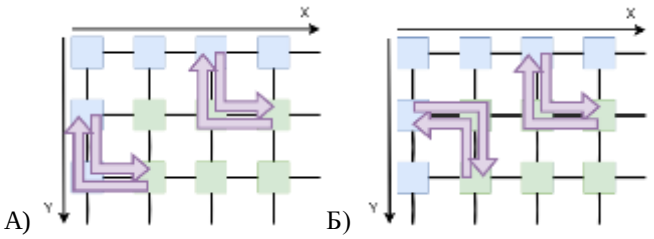
Среди ключевых архитектурных характеристик Mesh-сети, определяющих задержку обработки запросов, выделяются алгоритм маршрутизации, политика пакетной передачи и параметры емкости сети.

1) *Алгоритмы маршрутизации*

Современные исследования отмечают, что наиболее популярными алгоритмами маршрутизации на практике являются детерминированные алгоритмы [9], в частности алгоритм DOR (Dimension Order Routing). Популярность DOR объясняется его простотой, минимальными накладными расходами на реализацию и легкостью отладки. В то же время известно, что из-за несимметричности DOR (рис. 2А) пропускная способность вертикальных агентов при высоких значениях интенсивности может быть значительно ниже, чем у горизонтальных, что критично для вычислительных систем с распределенной памятью [10]. Для устранения этого недостатка был предложен более симметричный адаптивный алгоритм маршрутизации ADOR (рис. 2Б), позволяющий перераспределять нагрузку вертикальных агентов и тем самым повышать равномерность использования сети.

1) *Пакетная передача*

Важной характеристикой сетей на кристалле является политика передачи данных. Фундаментальные исследования показывают преимущество пакетной передачи по сравнению с посылочной (передача строго одной посылки данных на запрос) [11]. За счет



А) Б)

Рис. 2. Алгоритмы маршрутизации: (А) DOR и (Б) ADOR

упаковывания данных в пакеты существенно снижается трафик служебных сообщений, однако это требует дополнительной логики и накладных расходов. Среди классических подходов к управлению потоком пакетов наиболее распространен wormhole [12], при котором передача пакета начинается до его полного получения. В отличие от альтернативных решений — Store-and-Forward [13] и Virtual Cut-Through [14], требующих буферов размером не меньше пакета, — wormhole позволяет гибко регулировать объем аппаратных ресурсов, не накладывая жестких требований к буферизации. Несмотря на это, имеются работы, демонстрирующие преимущества Store-and-Forward и Virtual Cut-Through при специфических паттернах трафика [15, 16].

Еще одним важным аспектом является выбор размера пакета. Существуют исследования, отмечающие необходимость подбора оптимального размера в зависимости от параметров сети и характера трафика [17, 18]. В этом контексте особый интерес представляют сравнения пакетной и канальной коммутации. В случае пакетной коммутации маршрут прокладывается пошагово, тогда как канальная требует полного резервирования пути от источника к получателю [19]. Пакетная коммутация обеспечивает большую гибкость, а канальная — лучшую энергоэффективность [20]. При этом имеются данные о преимуществе канальной коммутации при очень больших размерах пакета [21], что подчеркивает важность выбора оптимального размера для конкретной архитектуры.

2) *Емкость сети*

Одним из главных факторов, определяющих емкость сети, является глубина буферов маршрутизаторов. Увеличение размеров буферов снижает задержку и повышает пропускную способность [22], однако сопровождается ростом затрат аппаратных ресурсов и энергопотребления. Это стимулирует интерес к архитектурам без буферов [23], где данные передаются на каждом такте без хранения. Такие решения сокращают площадь, но приводят к недетерминированности пути и усложняют отладку из-за возможных «отражений» пакетов.

Другим механизмом увеличения емкости является использование виртуальных каналов (ВК). Рост их числа улучшает производительность сети [24], но лишь до определенного предела, после которого дальнейшее

увеличение не дает выгоды, а только повышает энергопотребление [25]. Поэтому актуальным остается поиск оптимального соотношения между количеством ВК и глубиной буферов [26].

3) Управление виртуальными каналами

Стратегия распределения запросов по ВК также оказывает заметное влияние. Динамическая аллокация более гибкая и эффективна при средних нагрузках [27], в то время как статическая, закрепляющая канал за типом запроса, демонстрирует лучшие результаты при высоких нагрузках [28] и проще в реализации. Важную роль играет и внутренняя организация маршрутизатора. Предпочтительным с точки зрения задержки является однотактная обработка запросов [29], однако в современных системах с увеличивающимися размерами вычислительных ядер и объемами памяти требуется как минимум два такта [30]. В этом контексте активно обсуждаются решения с буферизацией не только во входных, но и в выходных каналах [31], а также независимая коммутация ВК, позволяющая параллельно обрабатывать несколько потоков [32].

Таким образом, задержка в сети определяется совокупным влиянием алгоритма маршрутизации, политики пакетной передачи и параметров емкости сети. Именно поэтому аналитическая модель, разрабатываемая в данной работе, должна уметь гибко варьировать ключевые архитектурные параметры: число ВК, глубину буферов, алгоритм маршрутизации и размер пакета.

B. Существующие аналитические модели

Аналитические методы для сетей на кристалле условно делятся на несколько направлений. Первое — гарантирующие подходы: оптимизация минимально гарантируемой пропускной способности при произвольных шаблонах трафика [33] и строгие оценки худшей задержки в wormhole-сетях с учетом арбитража и блокировок [34]. Эти методы важны для систем реального времени, но по своей природе консервативны: оценки заведомо завышают задержку, слабо отражают средние режимы и потому ограниченно пригодны для архитектурной оптимизации.

В блоке моделей, основанных на теории массового обслуживания, предлагаются два класса решений. С одной стороны, вводится упрощающее предположение о постоянном времени обслуживания маршрутизатора, что дает быструю и точную оценку при умеренных нагрузках [35], но теряет точность вблизи насыщения и не позволяет выявить узкие места в архитектуре маршрутизатора. С другой — постановки M/G/1/N с явным учетом блокировок и многоканальных структур, что улучшает точность оценки задержки и загрузки буферов [36], однако обычно опираются на упрощающие гипотезы о входном потоке, хуже масштабируются при сложной маршрутизации и не предоставляют разложение задержки по внутренним компонентам. Близкие по духу итеративные схемы вычисления средних задержек для wormhole-сетей учитывают конфликты каналов, глубину буферов и ВК [37], а также масштабируются на более общие постановки [7], но их точность остается чувствительной к принятым аппроксимациям и выбору числа итераций.

Повышение точности достигается в работах, где учитываются общие распределения межпакетных интервалов и времени обслуживания (например, GE/G/1/K), а зависимости между каналами описываются графами с явным учетом механизма обратного давления [38]. Эти модели точно предсказывают точку насыщения для широкого спектра нагрузок, но требуют калибровки распределений и параметров, усложняют вычислительную процедуру и, как правило, сохраняют жесткие допущения о маршрутизации и буферной организации.

Статистические и гибридные методы предлагают альтернативный компромисс между точностью и скоростью. Статистическая аппроксимация распределений задержек убирает необходимость итераций и ускоряет расчет при приемлемой точности на средних нагрузках [39], но теряет интерпретируемость и хуже работает у границы насыщения. Гибридные подходы, где аналитические формулы сочетаются с потактовой моделью конкуренции за каналы и буферы, позволяют учесть влияние позиции пакета в очереди, конфликты при передаче и блокировки, при этом вычисления выполняются заметно быстрее, чем при полномасштабном моделировании [40]. Однако подобные решения частично зависят от результатов потактового моделирования и сохраняют чувствительность к выбранным сценариям.

Сложные эксплуатационные сценарии описываются моделями для трафика с несколькими классами приоритетов [41]: приоритетная обработка потоков предсказуемо приводит к разной задержке для разных классов, и это дает возможность проектировщику выбирать баланс между задержкой высокоприоритетных и низкоприоритетных потоков в зависимости от требований приложения. Анализ гетерогенных сетей на кристалле показывает, что неоднородность частот, глубин буферов и пропускных способностей повышает задержки уже при умеренных нагрузках [42]. При этом модели нередко прибегают к дополнительным упрощающим допущениям и усложняются параметрически.

В литературе также встречаются специализированные модели, разработанные под конкретные микроархитектуры, отличающиеся особенностями коммутации. Эти решения демонстрируют высокую точность в условиях узкой архитектурной специализации [43, 44], но плохо переносятся на классические wormhole-сети общего назначения и потому ограниченно пригодны для раннего выбора архитектуры в типовой Mesh-топологии.

Наконец, подходы машинного обучения показывают, что регрессионные модели, обогащенные аналитическими признаками, способны повышать точность предсказаний задержек при значительном ускорении по сравнению с аналитическим моделированием [45]. Современный обзор подчеркивает растущую тенденцию объединять аналитику и методы машинного обучения для ускоренного исследования

проектного пространства и динамической адаптации сетей в процессе работы [46]. Однако подобные методы требуют обучающих данных, чувствительны к смене трафика и архитектуры и уступают аналитическим моделям по объяснимости результатов.

Существующие решения либо дают строгие гарантии и остаются консервативными [33, 34], либо упрощают внутреннюю структуру маршрутизаторов и теряют детализацию [7, 35, 36], либо узкоспециализированы под конкретные микроархитектуры [43, 44], либо требуют статистических допущений или зависят от обучающих данных и ко-симуляции [39, 40, 45, 46]. Эти ограничения мотивируют разработку модели задержек, сохраняющей высокую точность во всем диапазоне нагрузок вплоть до точки насыщения. Предлагаемый подход опирается на полностью аналитическое описание маршрутизаторов как систем M/M/1/N и поддерживает параметризацию по числу виртуальных каналов, глубине буферов, размеру пакета и типу маршрутизации. Модель предоставляет разложение средней задержки по внутренним компонентам, что делает результаты интерпретируемыми. В результате модель может служить инструментом для сопоставления архитектурных решений, объяснения наблюдаемых эффектов и обоснованного выбора параметров сети.

III. Аналитическая модель

Пусть X и Y – число столбцов и строк в двумерной Mesh-сети, тогда $T = X \cdot Y$ – общее число узлов. Каждый узел включает D входных и выходных портов, служащих для подключения, каждый порт включает P физических каналов, каждый физический канал состоит из V виртуальных каналов, каждый входной и выходной виртуальный канал оснащен FIFO буфером глубиной N (Рис. 3). Все обозначения, используемые далее при описании модели, сведены в таблице 1. В обозначениях для различения параметров входных и выходных буферов используются верхние индексы inp и out соответственно.

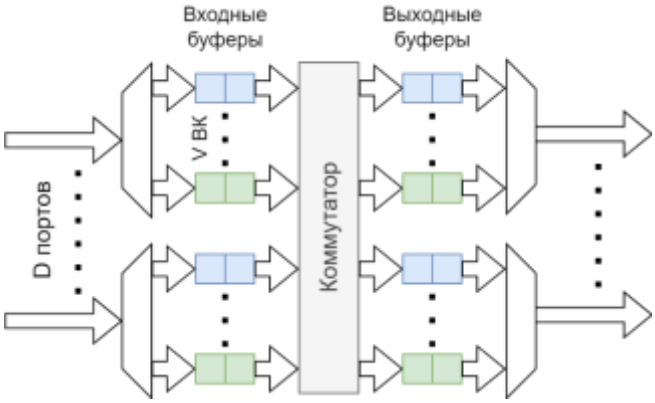


Рис. 3. Структура узла сети

При описании параметров и вычисляемых величин аналитической модели используется запись с нижними индексами, означающими следующее:

- Индекс $s \in \{0, \dots, T-1\}$ – узел источника запроса;
- Индекс $d \in \{0, \dots, T-1\}$ – узел получателя запроса;
- Индекс $r \in \{0, \dots, T-1\}$ – узел, через который происходит передача пакета;
- Индекс $i \in \{0, \dots, D-1\}$ – входной порт узла, по которому происходит передача пакета;
- Индекс $o \in \{0, \dots, D-1\}$ – выходной порт узла, по которому происходит передача пакета;
- Индекс $p \in \{0, \dots, P-1\}$ – физический канал порта, по которому происходит передача пакета;
- Индекс $v \in \{0, \dots, V-1\}$ – виртуальный канал физического канала, по которому происходит передача пакета.

Таблица 1. Параметры аналитической модели

Параметр	Описание
T	Число узлов в Mesh-сети
D	Число портов в узле Mesh-сети
P	Число физических каналов
V	Число виртуальных каналов
N	Глубина FIFO буфера ВК
$R_{sdrpvio}$	Индикатор маршрутизации
Λ^{p}_{sdpv}	Интенсивность формирования пакетов
L_{pv}	Размер пакета
$LA^{inp}_{pvsd}, LA^{out}_{pvsd}$	Средняя задержка обработки
D^{inp}_{rpvi}	Среднее число пакетов во входном буфере
$E(T^{inp}_{rpvi}), T^{inp}_{rpvio}$	Среднее время обработки во входном буфере
D^{out}_{rpvo}	Среднее число посылок в выходном буфере
$E(T^{out}_{rpvo}), T^{out}_{rpvo}$	Среднее время обработки в выходном буфере
$I^{p}_{rpvio}, I^{pinp}_{rpvi}$	Интенсивность прихода пакетов
$I^{f}_{rpvio}, I^{fout}_{rpvo}$	Интенсивность прихода посылок
$TC^{inp}_{rpvio}, TC^{out}_{rpvo}$	Время ожидания из-за коллизий
TB^{inp}_{rpvio}	Время ожидания из-за пакетов других входов
$PC^{inp}_{rpvio}, PC^{out}_{rpvo}$	Вероятность коллизии
$PR^{inp}_{rpvio}, PR^{out}_{rpvo}$	Вероятность наличия запроса
PCR^{inp}_{rpvio}	Вероятность коллизии при маршрутизации
$PCF^{inp}_{rpvo}, PCF^{out}_{rpvo}$	Вероятность переполнения целевого буфера
$U^{inp}_{rpvi}, U^{out}_{rpvo}$	Коэффициент использования буфера
$Pbuf^{inp}_{rpvi}(k)$, $Pbuf^{out}_{rpvo}(k)$	Вероятность нахождения в буфере ровно k запросов
PCS^{out}_{rpvo}	Вероятность коллизии при коммутации

Зададим маршрутизацию в Mesh-сети с помощью $R_{sdrpvio}$, принимающего значение 1 при совокупности следующих условий:

a) Существует трафик пакетов между s и d через r;
b) Передача осуществляется по p и v;
c) Маршрутизация производится с i на o;

и принимает значение 0 в противном случае. Зададим интенсивность формирования пакетов в сети с помощью Λ^{p}_{sdpv}, а размер пакетов в сети – с помощью L_{pv}.

Для оценки средней задержки передачи пакетов в Mesh-сети введем LA_{pvsd}:

$$LA_{pvsd} = LA^{inp}_{pvsd} + LA^{out}_{pvsd} + L_{pv} - 1,$$

где LA^{inp}_{pvsd} – средняя задержка обработки посылок во входных буферах маршрутизаторов;
LA^{out}_{pvsd} – средняя задержка обработки посылок в выходных буферах маршрутизаторов;

$(L_{pv}-1)$ – слагаемое, учитывающее конвейерный характер передачи пакета, состоящего из L_{pv} посылок.

LA^{inp}_{pvsd} моделирует задержку, в течение которой первая посылка пакета ожидает своей очереди на выход из буфера. В силу использования принципа передачи пакетов wormhole, при оценке задержки учитывается лишь первая посылка пакета, поскольку после того, как она выигрывает арбитраж и получает доступ к выходному каналу, все остальные посылки следуют за ней без дополнительных арбитражных задержек. При этом пакеты остальных входных буферов, претендующие на тот же выходной канал, будут заблокированы до тех пор, пока полностью не завершится передача текущего пакета. Таким образом, введя среднее число пакетов D^{inp}_{rpvi} в очереди и среднее времени обслуживания одного пакета $E(T^{inp}_{rpvi})$, задержка входных буферов получается суммированием по всем узлам на пути от s к d:

$$LA^{inp}_{pvsd} = \sum_{\substack{\forall r,i:\ \exists o\in\{0,\dots,D-1\}:\\ R_{sdrpvio}=1}} \left(D^{inp}_{rpvi}+1\right)\cdot E\left(T^{inp}_{rpvi}\right) - (L_{pv}-1).$$

Слагаемое $-(L_{pv}-1)$ в выражении выше исключает вклад в задержку всех посылок пакета, кроме первой, поскольку конвейерная передача пакета учитывается в итоговой средней задержке LA_{pvsd}.

LA^{out}_{pvsd} моделирует задержку посылки, ожидающей передачи на следующий маршрутизатор. Поскольку выходной буфер обрабатывает посылки в отрыве от пакетов, в которые они упакованы, введя среднее число посылок D^{out}_{rpvo} в очереди и среднее время обслуживания одной посылки $E(T^{out}_{rpvo})$, задержка обработки посылок в выходных буферах получается суммированием по всем маршрутизаторам на пути от источника s к получателю d:

$$LA^{out}_{pvsd} = \sum_{\substack{\forall r,o:\ \exists i\in\{0,\dots,D-1\}:\\ R_{sdrpvio}=1}} \left(D^{out}_{rpvo}+1\right)\cdot E\left(T^{out}_{rpvo}\right).$$

Для вычисления LA^{inp}_{pvsd} и LA^{out}_{pvsd} введем вспомогательные параметры, описывающие интенсивность поступления пакетов и посылок в каждый буфер узлов.

Интенсивность поступления пакетов в конкретный буфер определяется как:

$$I^{p}_{rpvio} = \sum_{s=0}^{T-1}\sum_{d=0}^{T-1} \Lambda^{p}_{sdpv}\cdot R_{sdrpvio}.$$

Поскольку каждый пакет состоит из L_{pv} посылок данных, интенсивность поступления посылок в тот же буфер можно выразить как:

$$I^{f}_{rpvio} = I^{p}_{rpvio}\cdot L_{pv}.$$

Суммируя по всем выходным портам при маршрутизации, интенсивность поступления пакетов во входные буферы выражается следующим образом:

$$I^{pinp}_{rpvi} = \sum_{o=0}^{D-1} I^{p}_{rpvio}.$$

Аналогично, суммируя по всем входным портам, получим интенсивность поступления посылок в выходные буферы:

$$I^{fout}_{rpvo} = \sum_{i=0}^{D-1} I^{f}_{rpvio}.$$

A. Задержки входных буферов

Переходя к анализу задержек во входных буферах маршрутизаторов, введем T^{inp}_{rpvio} – средняя продолжительность ожидания и обработки пакета на входе узла до момента начала его передачи на соответствующий выход. Данная величина определяется следующими тремя составляющими:

- TC^{inp}_{rpvio} – задержка, обусловленная коллизиями при маршрутизации пакета. Она возникает либо из-за конкуренции нескольких входов за один и тот же выход, либо в случае, если целевой выходной буфер переполнен и временно недоступен для передачи (рис. 4).
- TB^{inp}_{rpvio} – задержка, связанная с необходимостью завершения передачи предыдущих пакетов с других входов, направленных на тот же выходной порт. Поскольку передача пакета по принципу wormhole блокирует выход на время прохождения всех его посылок, вновь прибывший пакет может оказаться в ситуации, когда выход уже занят посылками ранее начавшегося пакета другого входа.
- L_{pv} — слагаемое, напрямую зависящее от размера пакета. Оно отражает необходимое количество тактов, чтобы передать все посылки данных пакета после того, как он получил доступ к выходу, при условии, что сеть работает в конвейерном режиме.

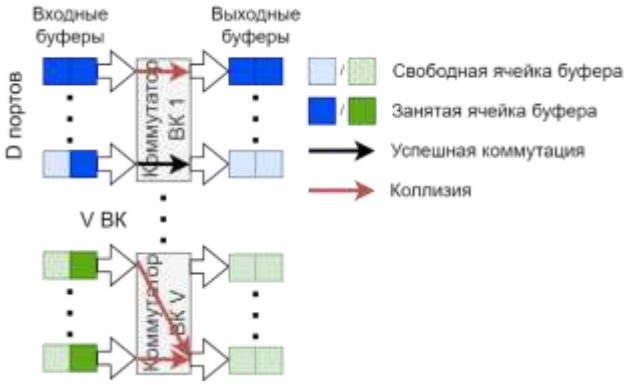


Рис. 4. Коллизии запросов во входных буферах

С учетом этих компонент общее выражение для T^{inp}_{rpvio} имеет вид:

$$T^{inp}_{rpvio} = TC^{inp}_{rpvio} + TB^{inp}_{rpvio} + L_{pv}.$$

Для формализации TC_{rpvio}^{inp} введем вспомогательные величины: PC_{rpvio}^{inp} и PR_{rpvio}^{inp}.

PC_{rpvio}^{inp} описывает вероятность того, что при попытке маршрутизации запроса возникнет коллизия. Эта вероятность учитывает два вида конфликтов: вызванные конкурентным доступом с других входов к тому же выходу PCR_{rpvio}^{inp}, а также конфликты, связанные с невозможностью немедленного доступа к выходному буферу PCF_{rpvo}^{inp}, в связи с его переполнением. Формально PC_{rpvio}^{inp} определяется как:

$$PC_{rpvio}^{inp} = 1 - \left(1 - PCR_{rpvio}^{inp}\right) \cdot \left(1 - PCF_{rpvo}^{inp}\right). \quad (1)$$

PR_{rpvio}^{inp} описывает вероятность того, что с i на o будет предпринята попытка передачи пакета. Эта вероятность может превышать интенсивность поступающего трафика I_{rpvio}^{p}, поскольку в условиях коллизий при маршрутизации запросы могут повторяться. Данная вероятность вычисляется по формуле геометрической прогрессии, на основе PC_{rpvio}^{inp}:

$$PR_{rpvio}^{inp} = I_{rpvio}^{p} \cdot \sum_{k=0}^{\infty} {PC_{rpvio}^{inp}}^{k} = \frac{I_{rpvio}^{p}}{1 - PC_{rpvio}^{inp}}. \quad (2)$$

Компонент PCR_{rpvio}^{inp} рассчитывается путем перебора возможных комбинаций одновременных запросов с разных входов на один и тот же выход. Эта величина выражается через суммирование по числу конкурирующих входов c и весовую функцию $\Delta^{inp}(i, o, c)$, которая описывает вероятность соответствующей конфигурации:

$$PCR_{rpvio}^{inp} = \sum_{c=1}^{D-1} \frac{c}{c+1} \cdot \Delta^{inp}(i, o, c). \quad (3)$$

$\Delta^{inp}(i, o, c)$ выражается через PR_{rpvio}^{inp} следующим образом:

$$\Delta^{inp}(i, o, c) = \sum_{\substack{k_1 + \ldots + k_{D-1} = c \\ \forall\, k_1, \ldots, k_{D-1} \in \{0,1\}}} \prod_{t=1}^{D-1} {PR_{rpvi_t o}^{inp}}^{k_t} \cdot \left(1 - PR_{rpvi_t o}^{inp}\right)^{1-k_t}, \quad (4)$$

где $\{i_1, \ldots, i_{D-1}\} = \{0, \ldots, D-1\} \backslash \{i\}$ – индексы всех входов узла, кроме i.

После вычисления полной вероятности конфликта PC_{rpvio}^{inp}, можно определить среднюю задержку из-за конфликтов TC_{rpvio}^{inp}, которая характеризует, сколько тактов в среднем занимает ожидание возможности начать передачу пакета при арбитраже. Это значение также моделируется как геометрическая прогрессия, поскольку при каждом конфликте попытка передать пакет повторяется:

$$TC_{rpvio}^{inp} = L_{pv} \cdot PC_{rpvio}^{inp} \cdot \sum_{k=0}^{\infty} {PC_{rpvio}^{inp}}^{k} = \frac{L_{pv} \cdot PC_{rpvio}^{inp}}{1 - PC_{rpvio}^{inp}}.$$

Это выражение показывает, что даже при умеренной вероятности конфликтов задержка может значительно возрасти, особенно для пакетов большого размера, поскольку по принципу wormhole весь пакет блокирует выход на протяжении всего времени своей передачи.

Задержка TB_{rpvio}^{inp} возникает из-за того, что в момент поступления нового пакета на вход i, выход o может уже быть занят передачей пакета, начатой с другого входа $k \neq i$. Эта задержка моделируется через суммирование ожидаемого остаточного времени передачи всех возможных пакетов, уже передающихся с других входов. Предполагается, что новый пакет поступает в случайный момент внутри интервала передачи уже идущего пакета, и, следовательно, в среднем должен ожидать половину оставшихся посылок. В итоге выражение для TB_{rpvio}^{inp} имеет следующий вид:

$$TB_{rpvio}^{inp} = \sum_{\substack{k=0 \\ k \neq i}}^{D-1} \left(I_{rpvko}^{f} \cdot \sum_{q=1}^{L_{pv}-1} \left(L_{pv} - q\right) \right) = \frac{L_{pv} - 1}{2} \cdot \sum_{\substack{k=0 \\ k \neq i}}^{D-1} I_{rpvko}^{f}.$$

Таким образом, после детального рассмотрения всех компонентов, влияющих на среднее время обработки пакета во входном буфере узла, вычислим агрегированные характеристики для каждого входного буфера. Для этого сначала усредним рассчитанное время обработки по всем выходам o, учитывая соответствующую интенсивность трафика:

$$E\left(T_{rpvi}^{inp}\right) = \frac{1}{I_{rpvi}^{pinp}} \cdot \sum_{o=0}^{D-1} I_{rpvio}^{p} \cdot T_{rpvio}^{inp}.$$

Рассматривая каждый входной буфер как систему массового обслуживания типа M/M/1/N, определим его коэффициент использования как:

$$U_{rpvi}^{inp} = I_{rpvi}^{pinp} \cdot E\left(T_{rpvi}^{inp}\right).$$

Выразим распределение вероятностей $Pbuf_{rpvi}^{inp}(k)$ заполненности входного буфера при текущей нагрузке:

$$Pbuf_{rpvi}^{inp}(k) = \frac{\left(1 - U_{rpvi}^{inp}\right) \cdot {U_{rpvi}^{inp}}^{k}}{1 - {U_{rpvi}^{inp}}^{N+1}}.$$

Тогда вероятность переполнения входного буфера равна $Pbuf_{rpvi}^{inp}(N)$, а его средняя глубина определяется как математическое ожидание по распределению:

$$D_{rpvi}^{inp} = \sum_{k=1}^{N} (k-1) \cdot Pbuf_{rpvi}^{inp}(k).$$

B. Задержки выходных буферов

В отличие от входных буферов, где анализ производится на уровне целых пакетов, моделирование выходных буферов удобно производить на уровне отдельных посылок. В отличие от входных буферов, где пакет занимает канал до тех пор, пока не будут переданы все посылки, выходные буферы обрабатывают посылки без привязки к пакетам. Таким образом, ключевой метрикой становится среднее время обработки одной посылки, поступающей в выходной буфер $E\left(T_{rpvo}^{out}\right)$:

$$E\left(T_{rpvo}^{out}\right) = T_{rpvo}^{out} = TC_{rpvo}^{out} + 1,$$

где TC_{rpvo}^{out} – средняя задержка, вызванная конфликтами за доступ к физическому каналу между посылками, поступающими из разных виртуальных каналов. Эта компонента учитывает не только коллизии, возникающие при мультиплексировании виртуальных каналов на один физический, но также и возможность временного блокирования передачи из-за переполнения соответствующего входного буфера соседнего маршрутизатора, в который направляется посылка (рис. 5).

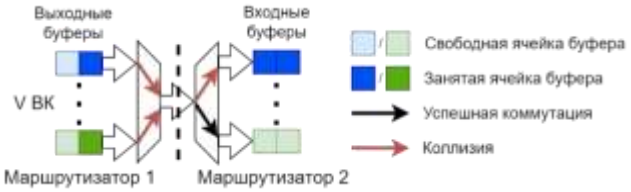


Рис. 5. Коллизии запросов в выходных буферах

Для учета обоих источников задержек в TC_{rpvo}^{out}, аналогично модели задержек входных буферов, введем вспомогательные величины: PC_{rpvo}^{out} и PR_{rpvo}^{out}.

PC_{rpvo}^{out} описывает вероятность коллизии при коммутации в выходных буферах и учитывает вероятность PCS_{rpvo}^{out} конфликта между виртуальными каналами за выход o, а также вероятность PCF_{rpvo}^{out} переполнения входного буфера соседнего маршрутизатора:

$$PC_{rpvo}^{out} = 1 - \left(1 - PCS_{rpvo}^{out}\right) \cdot \left(1 - PCF_{rpvo}^{out}\right). \quad (5)$$

PR_{rpvo}^{out} описывает вероятность наличия запроса на выходе o в виртуальном канале v. Эта вероятность может превышать интенсивность поступающего трафика I_{rpvo}^{fout}, поскольку в условиях коллизий между виртуальными каналами запросы могут повторяться. Данная вероятность вычисляется по формуле геометрической прогрессии, на основе PC_{rpvo}^{out}:

$$PR_{rpvo}^{out} = I_{rpvo}^{fout} \cdot \sum_{k=0}^{\infty} {PC_{rpvo}^{out}}^{k} = \frac{I_{rpvo}^{fout}}{1 - PC_{rpvo}^{out}}. \quad (6)$$

Компонент PCS_{rpvo}^{out} рассчитывается путем перебора возможных комбинаций одновременных запросов с разных виртуальных каналов на один и тот же выход. Эта величина выражается через суммирование по числу конкурирующих виртуальных каналов c и весовую функцию $\Delta^{out}(v, o, c)$, описывающую вероятность соответствующей конфигурации:

$$PCS_{rpvo}^{out} = \sum_{c=1}^{V-1} \frac{c}{c+1} \cdot \Delta^{out}(v, o, c). \quad (7)$$

$\Delta^{out}(v, o, c)$ выражается через PR_{rpvo}^{out} следующим образом:

$$\Delta^{out}(v, o, c) = \sum_{\substack{k_1 + \ldots + k_{V-1} = c \\ \forall\, k_1, \ldots, k_{V-1} \in \{0,1\}}} \prod_{t=1}^{V-1} {PR_{rpv_t o}^{out}}^{k_t} \cdot \left(1 - PR_{rpv_t o}^{out}\right)^{1-k_t}, \quad (8)$$

где $\{v_1, \ldots, v_{V-1}\} = \{0, \ldots, V-1\} \backslash \{v\}$ – индексы всех виртуальных каналов, кроме v.

Наконец, зная полную вероятность конфликта, можно вычислить задержку TC_{rpvo}^{out}, вызванную конфликтами, по формуле геометрической прогрессии:

$$TC_{rpvo}^{out} = PC_{rpvo}^{out} \cdot \sum_{k=0}^{\infty} {PC_{rpvo}^{out}}^{k} = \frac{PC_{rpvo}^{out}}{1 - PC_{rpvo}^{out}}.$$

Теперь, когда вычислены основные составляющие времени обработки посылок в выходных буферах узлов, можно завершить моделирование, описав поведение этих буферов как систем массового обслуживания типа M/M/1/N.

Коэффициент использования выходного буфера U_{rpvo}^{out} с интенсивностью поступления посылок I_{rpvo}^{fout} и средним временем их обработки $E\left(T_{rpvo}^{out}\right)$ определяется как:

$$U_{rpvo}^{out} = I_{rpvo}^{fout} \cdot E\left(T_{rpvo}^{out}\right).$$

Распределение вероятностей заполненности выходного буфера может быть выражено через U_{rpvo}^{out}:

$$Pbuf_{rpvo}^{out}(k) = \frac{\left(1 - U_{rpvo}^{out}\right) \cdot {U_{rpvo}^{out}}^{k}}{1 - {U_{rpvo}^{out}}^{N+1}}.$$

Тогда вероятность переполнения выходного буфера определяется как $Pbuf_{rpvo}^{out}(N)$, а его средняя глубина как математическое ожидание по распределению:

$$D_{rpvo}^{out} = \sum_{k=1}^{N} (k-1) \cdot Pbuf_{rpvo}^{out}(k).$$

Таким образом, завершив построение модели выходных буферов, получены все необходимые параметры для интеграции их в общую модель задержек.

IV. АЛГОРИТМ РАСЧЕТА АНАЛИТИЧЕСКОЙ МОДЕЛИ

Для оценки средней задержки в Mesh-сети необходимо вычислить среднюю задержку по всем возможным комбинациям трафика между s и d по каждому из каналов p и v, в соответствии с вышеописанной аналитической моделью, и произвести усреднение.

Входными данными для вычисления средней задержки LA_{pvsd} являются индикатор маршрутизации $R_{sdrpvio}$, интенсивность формирования пакетов Λ^{p}_{sdpv} и размер пакетов L_{pv}. Процесс вычисления LA_{pvsd} представлен в виде псевдокода (Алгоритм 1).

Алгоритм 1. Вычисление LA_{pvsd}

Вход: $R_{sdrpvio}$, Λ^{p}_{sdpv}, L_{pv}

Выход: LA_{pvsd}

```
1  function calc_LA(R_sdrpvio, p_lambda_sdpv, L_pv) begin
2      inp_PCF_rpvo ← init_with_zeros()
3      out_PCF_rpvo ← init_with_zeros()
4      p_I_rpvio ←
           calc_p_I_rpvio(R_sdrpvio, p_lambda_sdpv, L_pv)
5      f_I_rpvio ← calc_f_I_rpvio(p_I_rpvio, L_pv)
6      pinp_I_rpvi ← calc_pinp_I_rpvi(p_I_rpvio)
7      fout_I_rpvo ← calc_fout_I_rpvo(f_I_rpvio)
8      for iter from 0 to ITER_MAX begin
9          inp_PC_rpvio ←
               calc_inp_PC_rpvio(p_I_rpvio, inp_PCF_rpvo)
10         inp_TC_rpvio ← calc_inp_TC_rpvio(inp_PC_rpvio, L_pv)
11         inp_TB_rpvio ← calc_inp_TB_rpvio(p_I_rpvio, L_pv)
12         inp_T_rpvio ←
               calc_inp_T_rpvio(inp_TC_rpvio, inp_TB_rpvio, L_pv)
13         mean_inp_T_rpvi ← calc_mean_inp_T_rpvi(
               inp_T_rpvio, p_I_rpvio, pinp_I_rpvi)
14         (inp_D_rpvi, inp_Pbuf_full_rpvi) ←
               calc_MM1N(mean_inp_T_rpvi, pinp_I_rpvi)
15         out_PC_rpvo ←
               calc_out_PC_rpvo(fout_I_rpvo, out_PCF_rpvo)
16         out_TC_rpvo ← calc_out_TC_rpvo(out_PC_rpvo)
17         mean_out_T_rpvo ←
               calc_mean_out_T_rpvo(out_TC_rpvo)
18         (out_D_rpvo, out_Pbuf_full_rpvo) ←
               calc_MM1N(mean_out_T_rpvo, fout_I_rpvo)
19         inp_iter_diff ←
               inp_PCF_rpvo.update(out_Pbuf_full_rpvo)
20         out_iter_diff ← out_PCF_rpvo.update(inp_Pbuf_full_rpvi)
21         if inp_iter_diff < TOL && out_iter_diff < TOL begin
22             break
23         end
24     end
25     inp_LA_pvsd ←
           calc_inp_LA_pvsd(inp_D_rpvi, mean_inp_T_rpvi, L_pv)
26     out_LA_pvsd ←
           calc_out_LA_pvsd(out_D_rpvo, mean_out_T_rpvo)
27     LA_pvsd ← calc_LA_pvsd(inp_LA_pvsd, out_LA_pvsd, L_pv)
28     return LA_pvsd
29 end
```

В строках 4-7 алгоритма вычисляются вспомогательные параметры: I^{p}_{rpvio}, I^{f}_{rpvio}, I^{pinp}_{rpvi} и I^{fout}_{rpvo} соответственно. Далее (строки 8-24) выполняется подсчет параметров M/M/1/N систем: $E\left(T^{inp}_{rpvi}\right)$ и D^{inp}_{rpvi} для входных буферов, $E\left(T^{out}_{rpvo}\right)$ и D^{out}_{rpvo} для выходных буферов. Наконец, в строках 25–27 вычисляются средние задержки: LA^{inp}_{pvsd}, LA^{out}_{pvsd} и LA_{pvsd} соответственно.

Вычисление параметров M/M/1/N систем производится методом последовательного приближения, где на каждой итерации обновляются значения вероятностей переполнения буферов PCF^{inp}_{rpvo} и PCF^{out}_{rpvo}. В качестве начальных условий для данных вероятностей используются нулевые значения (строки 2-3). В рамках каждой итерации, в первую очередь, вычисляются величины для моделирования входных буферов (строки 9-14): PC^{inp}_{rpvio}, TC^{inp}_{rpvio}, TB^{inp}_{rpvio}, T^{inp}_{rpvio}, $E\left(T^{inp}_{rpvi}\right)$, D^{inp}_{rpvi} и $Pbuf^{inp}_{rpvi}(N)$. Затем производится вычисление величин, моделирующих выходные буферы (строки 15-18): PC^{out}_{rpvo}, TC^{out}_{rpvo}, $E\left(T^{out}_{rpvo}\right)$, D^{out}_{rpvo} и $Pbuf^{out}_{rpvo}(N)$. Наконец, с помощью полученных $Pbuf^{out}_{rpvo}(N)$ и $Pbuf^{inp}_{rpvi}(N)$, пересчитываются значения приближаемых PCF^{inp}_{rpvo} и PCF^{out}_{rpvo}, а также вычисляются их невязки – нормы изменений за текущую итерацию (строки 19-20). Цикл приближения завершается в случае, если значения каждой из невязок меньше наперед заданного параметра точности TOL, либо при исчерпании лимита итераций, заданного глобальным параметром $ITER\_MAX$.

Важно отметить, что вычисление PC^{inp}_{rpvio} производилось методом последовательного приближения с начальными нулевыми условиями (строка 9). Поскольку данная величина тесно связана с двумя вспомогательными PCR^{inp}_{rpvio} и PR^{inp}_{rpvio} в выражениях (1) – (4), ее невозможно аналитически выразить из них напрямую. Аналогично для PC^{out}_{rpvo} (строка 15), тесно связанной с PCS^{out}_{rpvo} и PR^{out}_{rpvo} в выражениях (5) – (8).

Для получения итогового значения средней задержки в сети из вычисленного LA_{pvsd}, введем индикаторы для каждого типа трафика: $WREQ_{pv}$ – запросы на запись, $WRESP_{pv}$ – подтверждения записи, $RREQ_{pv}$ – запросы на чтение и $RRESP_{pv}$ – прочитанные данные. Они принимают значение 1 в случае, если соответствующий вид трафика передается по каналам p и v, и принимают значение 0 в противном случае. Тогда средняя задержка для каждого из видов трафика может быть получена путем усреднения:

$$LA_{TRAF} = \left(\sum_{\substack{p,v: \\ TRAF_{pv}=1}} \sum_{s=0}^{T} \sum_{d=0}^{T} LA_{pvsd} \cdot \Lambda^{p}_{sdpv} \right) / \left(\sum_{\substack{p,v: \\ TRAF_{pv}=1}} \sum_{s=0}^{T} \sum_{d=0}^{T} \Lambda^{p}_{sdpv} \right),$$

где $TRAF$ – один из $\{WREQ, WRESP, RREQ, RRESP\}$. Наконец, итоговые значения средней задержки обработки запросов на запись и на чтения получаются суммированиями (9) и (10):

$$LA_{WRITE} = LA_{WREQ} + LA_{WRESP}, \tag{9}$$

$$LA_{READ} = LA_{RREQ} + LA_{RRESP}. \tag{10}$$

V. Описание сценария

В рамках данного исследования рассматривается Mesh-сеть размером 10×10 узлов, реализованная как часть архитектуры вычислительной системы с распределенной памятью. Агенты, формирующие запросы на чтение и запись, подключены к узлам по периметру сети, тогда как модули памяти размещены в центральной области (массив 8x8 внутренних узлов).

Каждый узел сети представляет собой маршрутизатор с пятью портами: четыре порта обеспечивают соединение с соседними узлами (север, юг, запад, восток), а пятый используется для подключения либо агента, либо модуля памяти.

Каждый порт маршрутизатора содержит два двунаправленных физических канала:

- Широкий канал, предназначенный для передачи данных как при чтении, так и при записи;
- Узкий канал, предназначенный для передачи управляющих сообщений: запросов на чтение и подтверждений записи.

Каждый физический канал расщепляется на множество виртуальных каналов, обеспечивающих логическую изоляцию потоков, а также играющих роль в предотвращении взаимоблокировок и увеличении пропускной способности. В рамках экспериментов число виртуальных каналов варьируется и составляет либо 2, либо 4 на каждый физический канал. Вход и выход каждого виртуального канала оснащен FIFO-очередью. Глубина очереди также варьируется в экспериментах: 2 либо 4 элемента.

Передача информации в сети осуществляется в виде пакетов, состоящих из одной или нескольких посылок. Каждый запрос на запись представляет собой пакет данных, содержащий от 1 до 4 посылок, в ответ на который отправляется единственное подтверждение. Каждый запрос на чтение сопровождается ответом в виде пакета, содержащего от 1 до 4 посылок данных. Размер пакета является одним из варьируемых параметров в исследовании.

В качестве алгоритма маршрутизации используется детерминированный подход, при этом в зависимости от конфигурации применяется либо классический алгоритм DOR, либо его модифицированная версия — гибкий DOR, меняющий направление маршрутизации для агентов, расположенных по вертикали.

Генерация трафика в сети моделируется как равномерное случайное воздействие. Исполнительные устройства с заданной интенсивностью формируют запросы на чтение и запись в произвольные узлы памяти. После обработки запроса модуль памяти отправляет обратно пакет с данными, либо подтверждение записи. Отношение между чтением и записью в потоке трафика фиксировано как 1:1. Интенсивность генерации запросов варьируется в зависимости от сценария эксперимента.

Основным критерием оценки эффективности сети является средняя задержка обработки запроса, измеряемая как суммарное время от момента формирования запроса до получения ответа (пакета данных или подтверждения).

Исследование включает серию экспериментов, направленных на изучение влияния ключевых архитектурных параметров на среднюю задержку обработки запросов.

VI. Численные результаты

Предложенная аналитическая модель была реализована на языке python и параметризирована с учетом описанного выше сценария. Модель использовалась для оценки средней задержки в сети в ходе ряда экспериментов. В проведенных экспериментах использовались несколько архитектурных конфигураций сети, представленных в таблице 2.

Конфигурации 1 и 4 были использованы для валидации аналитической модели путем сопоставления с потактовым симулятором. Конфигурации 1-3 использовались для выявления оптимальной стратегии к увеличению емкости сети. Конфигурации 1, 3 и 4 использовались для анализа причин роста эффективности при переходе от алгоритма DOR к ADOR. Конфигурации 1, 4-8 использовались для поиска оптимального размера пакета в сетях с алгоритмами DOR и ADOR.

Таблица 2. Параметры конфигураций в экспериментах

№	Алгоритм	V ВК	Глубина N	Размер пакета
1	DOR	2	2	1
2	DOR	2	4	1
3	DOR	4	2	1
4	ADOR	4	2	1
5	DOR	2	2	2
6	ADOR	4	2	2
7	DOR	2	2	4
8	ADOR	4	2	4

A. Валидация аналитической модели

Для подтверждения корректности предложенной аналитической модели была проведена серия сравнительных экспериментов с использованием потактового симулятора сетей на кристалле. В качестве эталонной модели применялся симулятор с открытым исходным кодом Noxim [47], используемый в академических и прикладных исследованиях архитектур сетей на кристалле.

Сопоставление проводилось для двух конфигураций Mesh-сети:

- сеть с алгоритмом DOR, использующим 2 виртуальных канала для избегания взаимоблокировок;
- сеть с алгоритмом ADOR, требующим 4 виртуальных канала.

Для обеспечения достоверности усредненных метрик каждый запуск потактового симулятора производился в течение 100 000 тактов, при этом для каждой заданной интенсивности выполнялось 100 независимых запусков. Это обеспечивало сглаживание случайных флуктуаций и позволило получить устойчивые оценки средней задержки в сети.

На рис. 6 приведен график средней задержки передачи запросов в зависимости от интенсивности генерации

трафика для обоих маршрутизаторов. Результаты аналитической модели показаны в сравнении с усредненными результатами, полученными в Noxim.

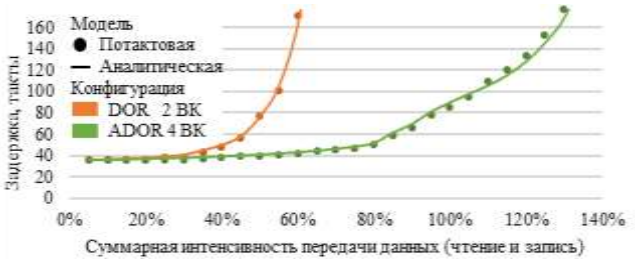


Рис. 6. Сравнение аналитической и потактовой моделей

Из сопоставления видно, что для обоих алгоритмов маршрутизации аналитическая модель демонстрирует хорошее согласование с симуляцией:

- отклонение не превышает 5% для всего диапазона заданных интенсивностей;
- при приближении к насыщению сети наблюдается характерный рост средней задержки;
- после определенного порога заданной интенсивности происходит неограниченное нарастание задержки, связанное с насыщением сети.

С точки зрения скорости моделирования следует отметить, что аналитическая модель обеспечила более чем в 100 раз более быстрое получение результатов по сравнению с потактовым симулятором, в силу отсутствия необходимости в усреднении результатов и отсутствия зависимости от числа моделируемых тактов.

B. Влияние емкости сети на задержку

Для анализа влияния емкости сети на среднюю задержку использовался классический алгоритм маршрутизации DOR. В исследовании сравнивались три конфигурации из таблицы 2:

- Конфигурация 1 – базовая: 2 виртуальных канала (ВК) на каждый физический канал, глубина буфера – 2 ячейки;
- Конфигурация 2 – увеличение глубины буферов вдвое (до 4 ячеек) при сохранении 2 ВК;
- Конфигурация 3 – увеличение числа ВК вдвое (до 4) при сохранении глубины буфера (2 ячейки).

Из полученных результатов средней задержки в зависимости от суммарной интенсивности генерации запросов (рис. 7) можно заключить, что удвоение глубины буферов (2) практически не изменяет интенсивность насыщения сети – как и в базовом варианте (1), насыщение наступает при интенсивности около 60%, после чего задержка неограниченно возрастает. Единственное заметное отличие – незначительное снижение средней задержки в диапазоне средних нагрузок 40–60%. Удвоение числа виртуальных каналов (3) оказывает гораздо более выраженный эффект: интенсивность насыщения возрастает до 80%, что свидетельствует о росте пропускной способности сети.

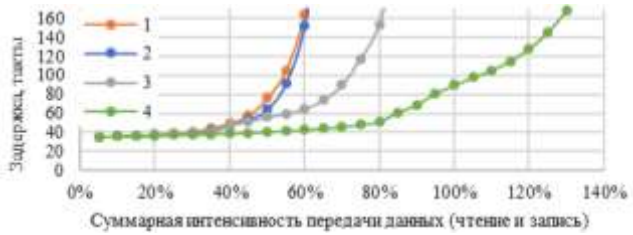


Рис. 7. Средняя задержка обработки запросов в Mesh-сети для конфигураций 1-4

Более детальный анализ структуры задержек, представленный на рис. 8, свидетельствует о том, что в базовой конфигурации (1) наибольший вклад в задержку вносят коллизии во входных буферах – на каждый из двух ВК приходится существенная нагрузка при маршрутизации. Распределение задержки для конфигурации с удвоенной глубиной буферов (2) совпадает с базовой в пределах 1%, что подтверждает отсутствие значимых изменений в структурах задержек. Для конфигурации с удвоенным числом ВК (3) преобладает задержка обработки в выходных буферах. С одной стороны, это обусловлено более равномерным распределением трафика по 4 ВК, снижающим число коллизий маршрутизации при обработке во входных буферах. С другой стороны, это подчеркивает рост коллизий коммутации при обработке в выходных буферах, возникающих из-за конкуренции между бóльшим числом ВК за доступ к физическому. В совокупности это подтверждает, что наиболее эффективным способом увеличения емкости сети является увеличение числа ВК, тогда как простое увеличение глубины буферов почти не влияет на интенсивность насыщения и структуру задержек.

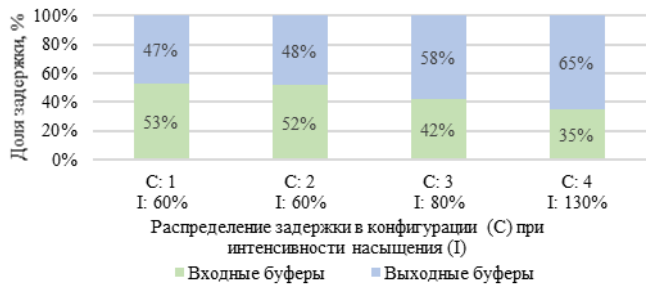


Рис. 8. Распределение средней задержки (~160 тактов) при насыщении Mesh-сети

C. Сравнение алгоритмов маршрутизации

Для сравнения алгоритмов DOR и ADOR были рассмотрены конфигурации с равными аппаратными ресурсами (4 ВК и глубина буферов 2), но различающиеся алгоритмом маршрутизации – конфигурации 3 и 4 из таблицы 2 соответственно.

Результаты сравнения средних задержек (рис. 7) показывают, что при равных ресурсах алгоритм ADOR обеспечивает более высокую пропускную способность сети: точка насыщения смещается с 80% (3) до 130% (4) суммарной интенсивности трафика.

Принимая во внимания результаты подраздела *B*, можно заключить, что ключевую роль в увеличении пропускной способности при переходе от классического DOR (1) к ADOR (4) играет именно алгоритм

маршрутизации, а не удвоение емкости сети. Распределение задержки между входными и выходными буферами в точке насыщения (рис. 8) дополнительно подтверждает этот вывод. Задержка, приходящаяся на входные буферы, минимальна у ADOR (4), что отражает снижение числа коллизий на этапе маршрутизации. При этом вклад выходных буферов в среднюю задержку максимален у ADOR (4) из-за роста конкуренции на этапе коммутации, возникающего при более полном использовании пропускной способности сети.

D. Оптимальный размер пакета в сети

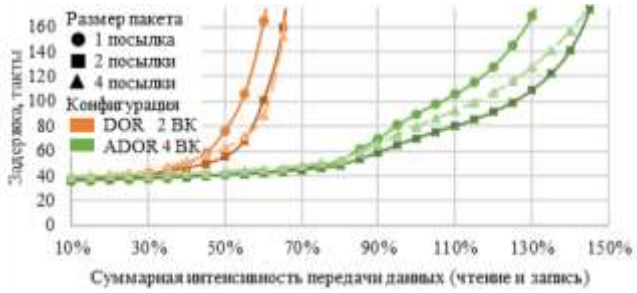
В рамках исследования по поиску оптимального размера пакета данных в сети рассматривался ряд конфигураций из таблицы 2, использующих размер 1, 2 и 4 посылки: конфигурации 1, 5 и 7 соответственно с алгоритмом маршрутизации DOR и конфигурации 4, 6 и 8 – с алгоритмом ADOR.

Из полученных результатов средней задержки обработки запросов на чтение (рис. 9А) и на запись (рис. 9Б) можно заключить, что при использовании пакетов данных, состоящих из двух посылок, достигается наилучшая пропускная способность сети при каждом из алгоритмов маршрутизации. Средняя задержка в сети с алгоритмом DOR при размере пакета 2 и 4 ниже, чем при размере пакета 1 при всех заданных интенсивностях, превышающих 45%, а точка насыщения смещается с 60% до 65% как в случае запросов на чтение, так и на запись. При этом средняя задержка при размере пакета 4 выше, чем при размере пакета 2 на интервале средних интенсивностей 40%-55% как для запросов на чтение, так и на запись. Средняя задержка в сети с алгоритмом ADOR при размере пакета 2 и 4 ниже, чем при размере пакета 1 при всех заданных интенсивностях, превышающих 85%. Насыщение сети, при этом, смещается с 130%:

- до 145% для запросов на чтение и до 150% для запросов на запись при размере пакета 2;
- до 145% для запросов на чтение и до 140% для запросов на запись при размере пакета 4.

При этом средняя задержка при размере пакета 4 выше, чем при размере пакета 2 при интенсивностях, превышающих 80%, как для запросов на чтение, так и на запись.

А)



Б)

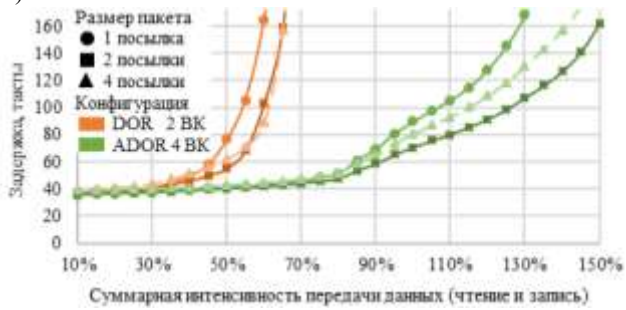


Рис. 9. Средняя задержка обработки запросов на (А) чтение и (Б) запись в Mesh-сети при различных размерах пакета

Более детальный анализ средней задержки обработки запросов на чтение и на запись в точке насыщения для каждой из конфигураций представлен на рис. 10А для алгоритма DOR и на рис. 10Б для алгоритма ADOR в виде двух типов распределений задержек. Из распределения по типу трафика (запросы/ответы) можно заключить, что пакетная передача существенно снижает долю служебных запросов в общую задержку. В конфигурациях с размером пакета 2 и 4 доля задержки, приходящаяся на передачу данных (запросы на запись и ответы при чтении), составляет 85%-89% для каждого из алгоритмов. При этом в конфигурации с размером пакета 1 вклады в задержку от передачи служебных запросов и данных равны.

Анализ распределения задержки между буферами показывает, что доля входных буферов растет с увеличением размера пакета – как для чтения, так и для записи, для каждого из алгоритмов. Это свидетельствует о том, что коллизии, связанные с пакетной передачей, вносят существенный вклад в общую задержку обработки запросов. Принимая во внимание результаты средней задержки (рис. 9), можно сделать вывод, что при размере пакета 4, выигрыш от сокращения служебного трафика компенсируется ростом коллизий и блокировок по механизму wormhole, что приводит к увеличению средней задержки.

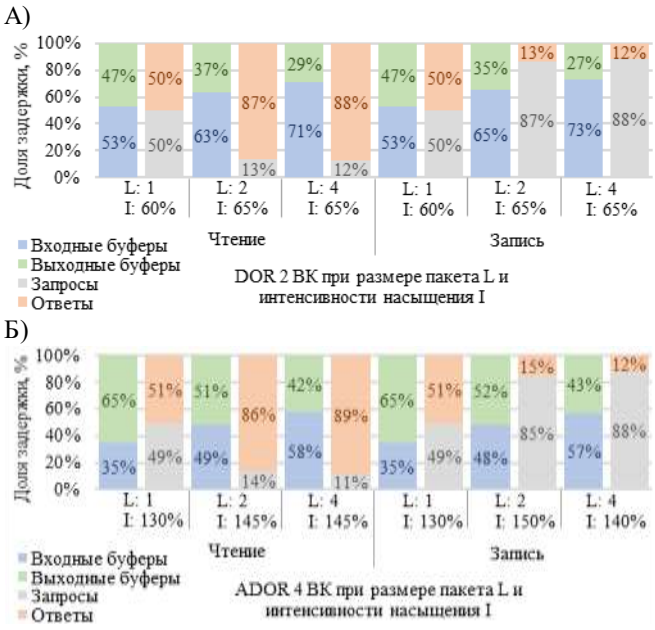


Рис. 10. Распределение средней задержки (~160 тактов) при насыщении Mesh-сети с алгоритмом (А) DOR и (Б) ADOR

VII. Заключение

В работе представлена аналитическая модель оценки задержек в Mesh-сети вычислительной системы с распределенной памятью. Маршрутизаторы трактуются как системы массового обслуживания типа M/M/1/N, а средняя задержка представляется в виде разложения по компонентам «вход/выход» и «запрос/ответ», что обеспечивает интерпретируемость результатов на уровне отдельных буферов и типов трафика. Модель реализована на Python и позволяет варьировать число виртуальных каналов, глубину буферов, размер пакета и вариант маршрутизации (DOR/ADOR), напрямую связывая архитектурные параметры с вкладом каждой подсистемы в суммарную задержку.

Корректность подхода подтверждена сопоставлением с потактовым симулятором: средняя ошибка по задержке не превышает порядка 5% в диапазоне нагрузок до насыщения, при этом вычисления выполняются примерно на два порядка быстрее за счет отсутствия длительных прогонов и многократных усреднений. Тем самым модель обеспечивает достаточную точность при существенно меньших затратах времени, что делает ее удобным инструментом для ранних этапов проектирования.

Предложенный подход может быть применен для количественного сравнения алгоритмов маршрутизации при равных аппаратных ресурсах, для оценки способов масштабирования емкости сети посредством изменения числа виртуальных каналов и глубины буферов, а также для обоснованного выбора размера пакета с учетом компромисса между снижением доли служебных сообщений и риском блокировок, характерных для wormhole-передачи.

Библиография

[1] Peccerillo, Biagio, et al. "A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives." Journal of Systems Architecture 129 (2022): 102561.
[2] McKee, Sally A. "Reflections on the memory wall." Proceedings of the 1st conference on Computing frontiers. 2004.
[3] Chen, Xiaowen. Efficient Memory Access and Synchronization in NoC-based Many-core Processors. Diss. KTH Royal Institute of Technology, 2019.
[4] Gholami, Amir, et al. "Ai and memory wall." IEEE Micro 44.3 (2024): 33-39.
[5] Milton, Jonathan, and Payman Zarkesh-Ha. "Impacts of topology and bandwidth on distributed shared memory systems." Computers 12.4 (2023): 86.
[6] Jain, Arpit, et al. "Smart Communication Using 2D and 3D Mesh Network-on-Chip." Intelligent Automation & Soft Computing 34.3 (2022).
[7] Kiasari, Abbas Eslami, Zhonghai Lu, and Axel Jantsch. "An analytical latency model for networks-on-chip." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 21.1 (2012): 113-123.
[8] Thomas, Marlin U. "Queueing systems. volume 1: Theory (leonard kleinrock)." SIAM Review 18.3 (1976): 512-514.
[9] Lallas, Efthymios N. "An Evaluation of Routing Algorithms in Traffic Engineering and Quality of Service Provision of Network on Chips." Engineering 13.1 (2021): 1-17.
[10] Khaidukov, Danila, and Alexandr Alekseev. "Routing Collision Mitigation Approaches in a Distributed Memory System on Chip". Nanoindustiya [Nanoindustry] 11s 18.135 (2025): 1298-1304.
[11] Kavaldjiev, Nikolay, and Gerard JM Smit. "A survey of efficient on-chip communications for soc." (2003).
[12] Uma, R., H. Sarojadevi, and V. Sanju. "Routing of flits in parallel input interface scenario in a generalized network-on-chip framework using wormhole flow control algorithm." Emerging Research in Computing, Information, Communication and Applications: ERCICA 2020, Volume 2. Singapore: Springer Singapore, 2021. 679-693.
[13] Dananjayan, P., and Karunakar Reddy Vanga. "Low Latency NoC Switch using Modified Distributed Round Robin Arbiter." Journal of Engineering Science & Technology Review 14.3 (2021).
[14] Kermani, Parviz, and Leonard Kleinrock. "Virtual cut-through: A new computer communication switching technique." Computer Networks (1976) 3.4 (1979): 267-286.
[15] Duato, Jose, et al. "A comparison of router architectures for virtual cut-through and wormhole switching in a NOW environment." Journal of Parallel and Distributed Computing 61.2 (2001): 224-253.
[16] Wang, Peng, et al. "A comprehensive comparison between virtual cut-through and wormhole routers for cache coherent Network on-Chips." IEICE Electronics Express 11.14 (2014): 20140496-20140496.
[17] Tedesco, Leonel P., Ney Calazans, and Fernando Moraes. "Buffer sizing for multimedia flows in packet-switching NoCs." Journal of Integrated Circuits and Systems 3.1 (2008): 46-56.
[18] Jin, Depeng, et al. "Evaluation and analysis of packet-length effect on networks-on-chip." Tsinghua Science & Technology 15.3 (2010): 288-293.
[19] Zhou, Xinbing, Peng Hao, and Dake Liu. "Pccnoc: Packet connected circuit as network on chip for high throughput and low latency socs." Micromachines 14.3 (2023): 501.
[20] Hao, Peng, et al. "PaCHNOC: Packet and Circuit Hybrid Switching NoC for Real-Time Parallel Stream Signal Processing." Micromachines 15.3 (2024): 304.
[21] Liu, Shaoteng, Axel Jantsch, and Zhonghai Lu. "Analysis and evaluation of circuit switched NoC and packet switched NoC." 2013 Euromicro Conference on Digital System Design. IEEE, 2013.
[22] Ghidini, Yan, et al. "Buffer depth and traffic influence on 3D NoCs performance." 2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP). IEEE, 2012.
[23] Parepalli, Ramanamma, Sanjeev Shama, and Mohan Kumar Naik. "Bufferless NoC router design for optical networks-on-chip." Journal of Optical Communications 0 (2024).
[24] Mirza-Aghatabar, Mohammad, et al. "An empirical investigation of mesh and torus NoC topologies under different routing algorithms and traffic models." 10th Euromicro conference on digital system design architectures, methods and tools (DSD 2007). IEEE, 2007.
[25] Nadi, M., M. H. Ghadiry, and M. K. Dermany. "The effect of number of virtual channel on NOC EDP." Journal of applied mathematics & informatics 2010 (2010): 539-551.
[26] Rezazad, Mostafa, and Hamid Sarbazi-Azad. "The effect of virtual channel organization on the performance of interconnection networks." 19th IEEE international parallel and distributed processing symposium. IEEE, 2005.

[27] Nicopoulos, Chrysostomos A., et al. "ViChaR: A dynamic virtual channel regulator for network-on-chip routers." 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06). IEEE, 2006.
[28] Shim, Keun Sup, et al. "Static virtual channel allocation in oblivious routing." 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip. IEEE, 2009.
[29] Fernandes, Ramon, et al. "OcNoC: Efficient one-cycle router implementation for 3d mesh network-on-chip." 2015 28th International Conference on VLSI Design. IEEE, 2015.
[30] Poluri, Pavan, and Ahmed Louri. "An improved router design for reliable on-chip networks." 2014 IEEE 28th International Parallel and Distributed Processing Symposium. IEEE, 2014.
[31] Papaphilippou, Philippos, and Thiem Van Chu. "Efficient deadlock avoidance for 2-D mesh NoCs that use OQ or VOQ routers." IEEE Transactions on Computers 73.5 (2024): 1414-1426.
[32] Rao, Supriya, et al. "Vix: Virtual input crossbar for efficient switch allocation." Proceedings of the 51st Annual Design Automation Conference. 2014.
[33] Seo, Daeho, et al. "Near-optimal worst-case throughput routing for two-dimensional mesh networks." 32nd International Symposium on Computer Architecture (ISCA'05). IEEE, 2005.
[34] Shi, Zheng, and Alan Burns. "Real-time communication analysis for on-chip networks with wormhole switching." Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008). IEEE, 2008.
[35] Nikitin, Nikita, and Jordi Cortadella. "A performance analytical model for network-on-chip with constant service time routers." Proceedings of the 2009 International Conference on Computer-Aided Design. 2009.
[36] Lai, Mingche, et al. "An accurate and efficient performance analysis approach based on queuing model for network on chip." Proceedings of the 2009 International Conference on Computer-Aided Design. 2009.
[37] Ogras, Umit Y., Paul Bogdan, and Radu Marculescu. "An analytical approach for network-on-chip performance analysis." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29.12 (2010): 2001-2013.
[38] Qian, Zhiliang, et al. "A comprehensive and accurate latency model for network-on-chip performance analysis." 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2014.
[39] Cohen, Itamar, Ori Rottenstreich, and Isaac Keslassy. "Statistical approach to networks-on-chip." IEEE Transactions on Computers 59.6 (2010): 748-761.
[40] Matoussi, Oumaima. "Noc performance model for efficient network latency estimation." 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021.
[41] Mandal, Sumit K., et al. "Analytical performance models for NoCs with multiple priority traffic classes." ACM Transactions on Embedded Computing Systems (TECS) 18.5s (2019): 1-21.
[42] Ben-Itzhak, Yaniv, Israel Cidon, and Avinoam Kolodny. "Delay analysis of wormhole based heterogeneous NoC." Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip. 2011.
[43] Bhattacharya, Debajit, and Niraj K. Jha. "Analytical modeling of the SMART NoC." IEEE Transactions on Multi-Scale Computing Systems 3.4 (2017): 242-254.
[44] Levitin, Lev B., and Yelena Rykalova. "An analytical model for virtual cut-through routing." 2019 28th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2019.
[45] Qian, Zhi-Liang, et al. "A support vector regression (SVR)-based latency model for network-on-chip (NoC) architectures." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35.3 (2015): 471-484.
[46] Zhang, Xiaoyun, et al. "A survey of machine learning for Network-on-Chips." Journal of Parallel and Distributed Computing 186 (2024): 104778.
[47] Catania, Vincenzo, et al. "Noxim: An open, extensible and cycle-accurate network on chip simulator." 2015 IEEE 26th international conference on application-specific systems, architectures and processors (ASAP). IEEE, 2015.

An Analytical Latency Estimation Approach in a Mesh Network in a Distributed Memory System on Chip

Danila Khaidukov, Aleksandr Alekseev

***Abstract*—We present an analytical latency model for a mesh Network-on-Chip with distributed memory in a system-on-chip, in which routers are modeled as M/M/1/N queues. The model accounts for collision probabilities arising at routing, at virtual-channel (VC) arbitration, and during wormhole transfer, and internal buffer-overflow probabilities. It supports comparison of routing schemes, number of VCs, buffer depth, and packet size, guiding early-stage decisions. Accuracy is validated against a cycle-accurate simulator: the error stays within 5% up to saturation while achieving ~100× speedup. Results are interpretable via decomposition of latency into contributions of input and output router buffers and into components associated with request and response traffic. Applying the model to routing, packet sizing, and capacity parameters yields three findings: (i) ADOR outperforms DOR due to the routing algorithm itself rather than added capacity (confirmed under equal hardware resources); (ii) packet-based transfer outperforms single-payload transfer, with an optimal packet size (two flits in our experiments); and (iii) when scaling capacity, increasing the number of VCs is more effective than increasing buffer depth.**

***Keywords*—analytical model, computing system, distributed memory, network on chip, mesh.**

REFERENCES

[1] Peccerillo, Biagio, et al. "A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives." Journal of Systems Architecture 129 (2022): 102561.
[2] McKee, Sally A. "Reflections on the memory wall." Proceedings of the 1st conference on Computing frontiers. 2004.
[3] Chen, Xiaowen. Efficient Memory Access and Synchronization in NoC-based Many-core Processors. Diss. KTH Royal Institute of Technology, 2019.
[4] Gholami, Amir, et al. "Ai and memory wall." IEEE Micro 44.3 (2024): 33-39.
[5] Milton, Jonathan, and Payman Zarkesh-Ha. "Impacts of topology and bandwidth on distributed shared memory systems." Computers 12.4 (2023): 86.
[6] Jain, Arpit, et al. "Smart Communication Using 2D and 3D Mesh Network-on-Chip." Intelligent Automation & Soft Computing 34.3 (2022).
[7] Kiasari, Abbas Eslami, Zhonghai Lu, and Axel Jantsch. "An analytical latency model for networks-on-chip." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 21.1 (2012): 113-123.
[8] Thomas, Marlin U. "Queueing systems. volume 1: Theory (leonard kleinrock)." SIAM Review 18.3 (1976): 512-514.
[9] Lallas, Efthymios N. "An Evaluation of Routing Algorithms in Traffic Engineering and Quality of Service Provision of Network on Chips." Engineering 13.1 (2021): 1-17.
[10] Khaidukov, Danila, and Aleksandr Alekseev. "Routing Collision Mitigation Approaches in a Distributed Memory System on Chip". Nanoindustiya [Nanoindustry] 11s 18.135 (2025): 1298-1304.
[11] Kavaldjiev, Nikolay, and Gerard JM Smit. "A survey of efficient on-chip communications for soc." (2003).
[12] Uma, R., H. Sarojadevi, and V. Sanju. "Routing of flits in parallel input interface scenario in a generalized network-on-chip framework using wormhole flow control algorithm." Emerging Research in Computing, Information, Communication and Applications: ERCICA 2020, Volume 2. Singapore: Springer Singapore, 2021. 679-693.
[13] Dananjayan, P., and Karunakar Reddy Vanga. "Low Latency NoC Switch using Modified Distributed Round Robin Arbiter." Journal of Engineering Science & Technology Review 14.3 (2021).
[14] Kermani, Parviz, and Leonard Kleinrock. "Virtual cut-through: A new computer communication switching technique." Computer Networks (1976) 3.4 (1979): 267-286.
[15] Duato, Jose, et al. "A comparison of router architectures for virtual cut-through and wormhole switching in a NOW environment." Journal of Parallel and Distributed Computing 61.2 (2001): 224-253.
[16] Wang, Peng, et al. "A comprehensive comparison between virtual cut-through and wormhole routers for cache coherent Network on-Chips." IEICE Electronics Express 11.14 (2014): 20140496-20140496.
[17] Tedesco, Leonel P., Ney Calazans, and Fernando Moraes. "Buffer sizing for multimedia flows in packet-switching NoCs." Journal of Integrated Circuits and Systems 3.1 (2008): 46-56.
[18] Jin, Depeng, et al. "Evaluation and analysis of packet-length effect on networks-on-chip." Tsinghua Science & Technology 15.3 (2010): 288-293.
[19] Zhou, Xinbing, Peng Hao, and Dake Liu. "Pccnoc: Packet connected circuit as network on chip for high throughput and low latency socs." Micromachines 14.3 (2023): 501.
[20] Hao, Peng, et al. "PaCHNOC: Packet and Circuit Hybrid Switching NoC for Real-Time Parallel Stream Signal Processing." Micromachines 15.3 (2024): 304.
[21] Liu, Shaoteng, Axel Jantsch, and Zhonghai Lu. "Analysis and evaluation of circuit switched NoC and packet switched NoC." 2013 Euromicro Conference on Digital System Design. IEEE, 2013.
[22] Ghidini, Yan, et al. "Buffer depth and traffic influence on 3D NoCs performance." 2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP). IEEE, 2012.
[23] Parepalli, Ramanamma, Sanjeev Shama, and Mohan Kumar Naik. "Bufferless NoC router design for optical networks-on-chip." Journal of Optical Communications 0 (2024).
[24] Mirza-Aghatabar, Mohammad, et al. "An empirical investigation of mesh and torus NoC topologies under different routing algorithms and traffic models." 10th Euromicro conference on digital system design architectures, methods and tools (DSD 2007). IEEE, 2007.
[25] Nadi, M., M. H. Ghadiry, and M. K. Dermany. "The effect of number of virtual channel on NOC EDP." Journal of applied mathematics & informatics 2010 (2010): 539-551.
[26] Rezazad, Mostafa, and Hamid Sarbazi-Azad. "The effect of virtual channel organization on the performance of interconnection networks." 19th IEEE international parallel and distributed processing symposium. IEEE, 2005.
[27] Nicopoulos, Chrysostomos A., et al. "ViChaR: A dynamic virtual channel regulator for network-on-chip routers." 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06). IEEE, 2006.
[28] Shim, Keun Sup, et al. "Static virtual channel allocation in oblivious routing." 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip. IEEE, 2009.
[29] Fernandes, Ramon, et al. "OcNoC: Efficient one-cycle router implementation for 3d mesh network-on-chip." 2015 28th International Conference on VLSI Design. IEEE, 2015.
[30] Poluri, Pavan, and Ahmed Louri. "An improved router design for reliable on-chip networks." 2014 IEEE 28th International Parallel and Distributed Processing Symposium. IEEE, 2014.
[31] Papaphilippou, Philippos, and Thiem Van Chu. "Efficient deadlock

avoidance for 2-D mesh NoCs that use OQ or VOQ routers." IEEE Transactions on Computers 73.5 (2024): 1414-1426.
[32] Rao, Supriya, et al. "Vix: Virtual input crossbar for efficient switch allocation." Proceedings of the 51st Annual Design Automation Conference. 2014.
[33] Seo, Daeho, et al. "Near-optimal worst-case throughput routing for two-dimensional mesh networks." 32nd International Symposium on Computer Architecture (ISCA'05). IEEE, 2005.
[34] Shi, Zheng, and Alan Burns. "Real-time communication analysis for on-chip networks with wormhole switching." Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008). IEEE, 2008.
[35] Nikitin, Nikita, and Jordi Cortadella. "A performance analytical model for network-on-chip with constant service time routers." Proceedings of the 2009 International Conference on Computer-Aided Design. 2009.
[36] Lai, Mingche, et al. "An accurate and efficient performance analysis approach based on queuing model for network on chip." Proceedings of the 2009 International Conference on Computer-Aided Design. 2009.
[37] Ogras, Umit Y., Paul Bogdan, and Radu Marculescu. "An analytical approach for network-on-chip performance analysis." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29.12 (2010): 2001-2013.
[38] Qian, Zhiliang, et al. "A comprehensive and accurate latency model for network-on-chip performance analysis." 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2014.
[39] Cohen, Itamar, Ori Rottenstreich, and Isaac Keslassy. "Statistical approach to networks-on-chip." IEEE Transactions on Computers 59.6 (2010): 748-761.
[40] Matoussi, Oumaima. "Noc performance model for efficient network latency estimation." 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021.
[41] Mandal, Sumit K., et al. "Analytical performance models for NoCs with multiple priority traffic classes." ACM Transactions on Embedded Computing Systems (TECS) 18.5s (2019): 1-21.
[42] Ben-Itzhak, Yaniv, Israel Cidon, and Avinoam Kolodny. "Delay analysis of wormhole based heterogeneous NoC." Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip. 2011.
[43] Bhattacharya, Debajit, and Niraj K. Jha. "Analytical modeling of the SMART NoC." IEEE Transactions on Multi-Scale Computing Systems 3.4 (2017): 242-254.
[44] Levitin, Lev B., and Yelena Rykalova. "An analytical model for virtual cut-through routing." 2019 28th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2019.
[45] Qian, Zhi-Liang, et al. "A support vector regression (SVR)-based latency model for network-on-chip (NoC) architectures." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35.3 (2015): 471-484.
[46] Zhang, Xiaoyun, et al. "A survey of machine learning for Network-on-Chips." Journal of Parallel and Distributed Computing 186 (2024): 104778.
[47] Catania, Vincenzo, et al. "Noxim: An open, extensible and cycle-accurate network on chip simulator." 2015 IEEE 26th international conference on application-specific systems, architectures and processors (ASAP). IEEE, 2015.