Моделирование алгоритмов маршрутизации в сетях на кристалле

Ладыженский Ю.В., Мирецкая В.А.
Донецкий национальный технический университет
ly@cs.dgtu.donetsk.ua, vmiretskaya@gmail.com

Abstract

Ladyzhensky Y., Miretskaya V. Simulation of routing algorithms for network on chip. *Hardware implementation model of routing algorithm AntNet is presented in this paper. Simulation system for network on chip is presented. Behavior of AntNet was explored in heavy packet traffic and compared with other routing algorithm XY.*

Введение

Использование сетей на кристалле (Network-on-Chip - NOC) получило широкое распространение.

Среди факторов интенсивного развития сетей на кристалле можно выделить: рост сложности разрабатываемых систем, потребность в создании высокопроизводительных, гибких, масштабируемых, программируемых микропроцессорных схем и платформ для многократного использования.

При разработке NOC следует минимизировать стоимость аппаратного обеспечения, то есть необходимо минимизировать потребление энергии и площадь микросхемы.

Актуальной является задача разработки эффективных алгоритмов маршрутизации для сетей на кристалле. Маршрутизация в сетях на кристалле характеризуется задержкой пакетов в сети и пропускной способностью. Маршрутизация считается хорошей, если удается уменьшить задержку пакетов, увеличить пропускную способность, исключить перегруженные узлы.

Моделирование является одним из основных этапов в разработке алгоритмов маршрутизации. Ввиду сложности объекта исследования, целесообразно использование имитационного моделирования.

В статье предлагается модель сети на кристалле, использующей алгоритм маршрутизации AntNet. Представлена разрабатываемая система имитационного моделирования сетей на кристалле. Экспериментально показано, что алгоритм AntNet маршрутизирует потоки данных таким образом, что в сети не возникают перегруженные узлы.

Архитектура сетей на кристалле

В сети на кристалле традиционная шинная архитектура заменяется сетью похожей на сеть Internet. Данные в такой сети передаются в виде пакетов.

Сеть включает маршрутизаторы, соединительные проводники и PE-блоки (Process Element – вычислительный процессор), которые обмениваются информацией по сети.

Топологии сетей на кристалле

Топология сетей на кристалле может быть регулярной и нерегулярной [1].

Наиболее распространены среди однородных топологий:

– Топология Сетка (рис. 1а)
– Топология Тор (рис. 1б)
– Топология Складчатый тор (рис. 1в)
– Топология Гиперкуб (рис. 1г)

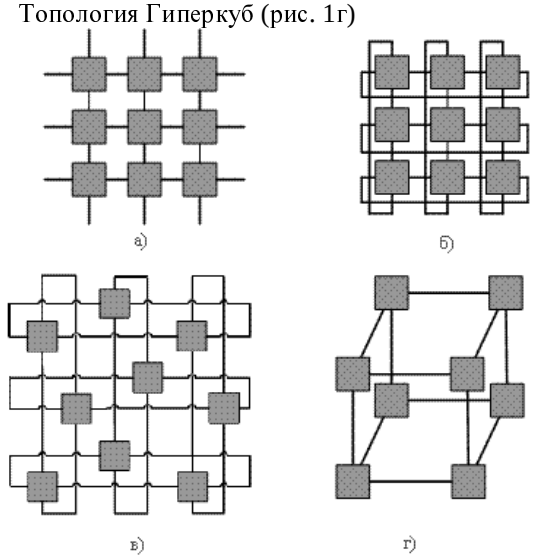

Рисунок 1 – Однородные топологии NOC

Для решения специфических задач используются сети с нерегулярной топологией [2].

Проблемы маршрутизации в сетях на кристалле

Основной задачей алгоритмов маршрутизации является передача данных от

процессора-источника к процессору-приемнику, максимизируя при этом производительность сети.

Для управления сетевым потоком могут использоваться различные режимы маршрутизации. В NOC-системах предпочтительно использовать маршрутизацию с коммутацией каналов (wormhole routing) [1].

В этом режиме маршрутизации при передаче, пакет делится на небольшие части, называемые flit-пакетами (flow control digit, быстрый пакет) [3]. Функция маршрутизации применяется только к первому флиту (заголовку) пакета, остальные же флиты передаются по установившемуся пути. Такая процедура передачи позволяет минимизировать использование ресурсов и минимизировать задержку пакетов, так как основная работа происходит с первым флитом пакета.

Алгоритмы маршрутизации можно классифицировать на статические и адаптивные. В статических системах маршрутизации маршрут, который проходит пакет, определяется только его источником и приемником, без учета текущего состояния сети. Адаптивная маршрутизация учитывает изменение текущего состояния сети. Алгоритмы могут определять путь в соответствии с установленным критерием оптимальности, обычно это минимальная стоимость пути.

Из общих проблем маршрутизации для сетей на кристалле характерны [1]:

1) взаимоблокировка (deadlock);
2) активный тупик (livelock);
3) зависание процесса (starvation).

Взаимоблокировка – это такое состояние маршрутизаторов, когда они ожидают освобождения ресурсов друг друга для отправки пакетов. На рисунке 2 показан пример взаимоблокировки четырех маршрутизаторов.

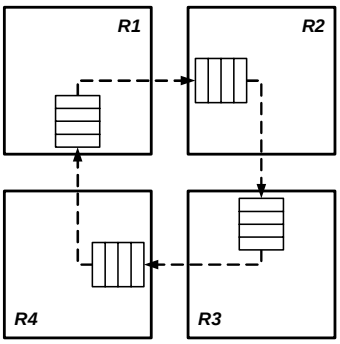

Рисунок 2 – Взаимоблокировка

Входные очереди маршрутизаторов, показанные на рисунке, заполнены, следовательно, помещение в эти очереди новых пакетов запрещено. Стрелками показаны направления передачи пакетов из входных очередей. Пакеты не могут быть отправлены по этим направлениям, т.к. соответствующие входные очереди принимающих маршрутизаторов заполнены. Отправка пакетов заблокирована.

Активные тупики возникают если пакет движется около своего узла назначения, но не достигает его. На рисунке 3 показан пример активного тупика, в котором пакет пытается достичь своего узла назначения R5.

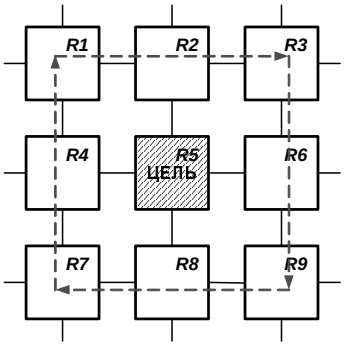

Рисунок 3 – Активный тупик

Существует несколько способов исключения активных тупиков.

Счетчик TTL (Time To live) считает как долго пакет находится в сети. Когда счетчик достигает некоторого определенного значения, пакет удаляется из сети. Другой способ - это назначить пакетам приоритет в соответствии с их возрастом. Старейший пакет всегда получит больший приоритет и будет маршрутизирован.

Использование различных приоритетов может привести к ситуации, когда некоторые пакеты с самым низким приоритетом никогда не достигнут узла назначения. Это происходит, когда пакеты с более высоким приоритетом постоянно получают ресурсы. Тогда возникает зависание процесса передачи данных. Зависание процесса можно избежать, используя справедливые алгоритмы маршрутизации [1] или резервируя часть пропускной способности исключительно для пакетов с низким приоритетом.

В сетях могут возникать точки скученности данных (hot-spot) [4] – это узлы в которых концентрируется входной и выходной трафик с высокой интенсивностью. Появление точек скученности приводит к перегрузке некоторых участков сети и нагреву микросхемы. Алгоритмы статической маршрутизации не могут находить новые маршруты, которые позволили бы перенаправлять потоки данных в обход точек скученности.

Алгоритмы маршрутизации для сетей на кристалле

Для сетей на кристалле наиболее часто используют следующие алгоритмы маршрутизации:

- Алгоритм XY и его модификации [2, 4]
- Алгоритм Odd-Even [1,4]
- Алгоритм DyAD [1,4, 5]

Алгоритм XY

XY-маршрутизация относится к статической маршрутизации. При использовании алгоритма XY маршрутизация данных осуществляется сначала в горизонтальном направлении (по координате X), а затем в вертикальном (по координате Y) (см. рис. 4).

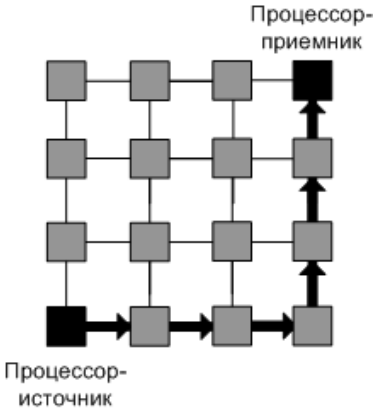

Рисунок 4 – XY -маршрутизация

Алгоритм XY хорошо подходит, например, для сетей с регулярной топологией сетка и тор [2]. XY никогда не приводит к взаимоблокировкам или активным тупикам. Однако при таком алгоритме трафик не распределяется равномерно в сети, а сосредотачивается, в основном, в ее середине.

Применение в NOC статической маршрутизации более предпочтительно. Однако, использование данного типа маршрутизации в сетях с нерегулярной топологией, нежелательно [2]

Алгоритм AntNet для сети на кристалле

Алгоритм AntNet [6] является адаптивным алгоритмом маршрутизации, метод поиска путей которого основан на поведении муравьев в природе. В нем предусмотрено исследование состояния сети с использованием агентов (муравьев), которые используют вероятностные правила выбора маршрутов. Для маршрутизации пакетов данных используются вероятностные таблицы маршрутизации.

В [7] приведены результаты экспериментального исследования, показывающие, что AntNet позволяет снизить вероятность перегрузки и появления точек скученности в компьютерной сети.

Рассмотрим подробно реализацию этого алгоритма в сетях на кристалле.

Для упрощения использования вероятностей в аппаратной реализации значения вероятности кодируются целыми числами. Пусть *Dir* – количество направлений, по которым маршрутизатор может отправлять пакеты другим маршрутизаторам, тогда вероятность *P=[0..1]* можно закодировать целыми числами *[0..Dir]*, т.е. код $c \in [0..Dir]$ соответствует вероятности $P=c/Dir$. Для сетей, в которых маршрутизаторы имеют неодинаковое число соединений, величина *Dir* вычисляется как

$$Dir = \max_{k=1..K}(N_k) \quad (1)$$

где K – количество узлов в сети; N_k – количество узлов соседних с k узлом.

При движении муравья для выбора следующего узла используется формула [6]:

$$P_{nd}^{'} = \frac{P_{nd} + \alpha \cdot l_n}{1 + \alpha(N_k - 1)} \quad (2)$$

где P_{nd} – значение из таблицы маршрутизации, определяющее вероятность достижения из текущего узла k узла d через узел n; α – весовой коэффициент, $\alpha \in [0.2;0.5]$; l_n – состояние очереди пакетов между текущим узлом и узлом n.

Пусть α=1/3, тогда учитывая определенный выше параметр *Dir*, формулу (2) можно привести к виду:

$$P'_{jd} = \frac{3 \times P_{jd} + L_j}{2 + N_k} = \frac{3 \times P_{jd} + L_j}{2 + Dir} \quad (3)$$

Для обновления таблиц маршрутизации в оригинальном алгоритме AntNet используются формулы [6]:

$$P_{fd'} := P_{fd'} + r \cdot (1 - P_{fd'}) \quad (4)$$

$$P_{nd'} := P_{nd'} - r \cdot P_{nd'}, \quad n \in N_k, n \neq f \quad (5)$$

где $r \in (0,1]$ – коэффициент стабилизации, аналог феромонов. В [4] для вычисления коэффициента стабилизации предложено использовать состояние перегрузки узла – CS (Congestion Status). Значение CS для узла k определяется как сумма Флагов Перегрузки (CF):

$$CS = \frac{Dir}{N_k} \sum_{i=1}^{N_k} CF_i \quad (6)$$

где Dir/N_k – масштабный коэффициент; при вычислении CS используется округление до ближайшего целого. Каждый из флагов CF соответствует своему направлению в сети. Масштабный коэффициент необходим, т.к. значение CS должно варьироваться в том же диапазоне, что и значение вероятности $CS \in [0..Dir]$.

Коэффициент стабилизации *r=1–CS/Dir*. Таким образом, формулы (4) и (5) могут быть модифицированы:

$$P'_{fd} = P_{fd} + \frac{(Dir - CS) \cdot (Dir - P_{fd})}{Dir} = Dir - CS + CS \cdot P_{fd} \cdot Dir^{-1} \quad (7)$$

$$P'_{nd} = P_{nd} - \frac{(Dir - CS) \cdot P_{nd}}{Dir} = CS \cdot P_{nd} \cdot Dir^{-1}, \forall n \neq f \quad (8)$$

F-Муравьи выпускаются в сеть периодически каждым узлом. При выборе узла назначения в [6] используется концепция «популярного узла», т.е. такого узла, в который чаще всего следовали пакеты данных.

В оригинальном AntNet в F-муравьях сохраняется путь их движения как список идентификаторов узлов, которые прошел муравей. Этот список используется B-муравьями для возвращения в узел, выпустивший F-муравья. Для уменьшения длины пакетов F-муравьев, а следовательно и времени, затрачиваемого на их обработку, предлагается сохранять в списке коды входных портов, через которые маршрутизатор принимал F-муравьи. Для хранения идентификатора узла требуется $\lceil Log_2(\mathrm{K}) \rceil$ бит памяти, а для хранения кода входного порта $\lceil Log_2(\mathrm{D}ir) \rceil$. K>Dir, следовательно $\lceil Log_2(\mathrm{D}ir) \rceil < \lceil Log_2(K) \rceil$. Хранение кодов входных портов упрощает схему вычисления маршрута для B-муравьев, т.к. нет необходимости по идентификатору узла вычислять номер выходного порта. Номера входного порта и выходного порта, соединенных с одним и тем же узлом, равны.

Разработанная модификация алгоритма AntNet отличается от модификации, приведенной в [4] тем, что ее можно применять в нерегулярных сетях на кристалле с произвольным числом портов у процессорного элемента. Хранение кодов входных портов вместо идентификаторов позволяет уменьшить длину пакетов муравьев и сократить время вычисления маршрута при обработке B-муравьев.

Формат пакетов муравьев

Основной целью F-муравья является сбор информации о состоянии сети. По прибытии в процессор-назначения, F-муравей становится B-муравьем и, двигаясь, в обратном направлении, обновляет таблицы маршрутизации. Структура флитов пакетов муравьев представлена в таблицах 1 и 2.

Таблица 1. Формат заголовка F-муравья и последнего флита B-муравья

Название	Размер, бит	Описание
IsFAnt	1	0 – B-Ant, 1 – F-Ant
Dst	16	Узел-назначение F-муравья
CS	3	Congestion status значение - 0…4
TTL	6	Time to Live
Dir1… Dir2	по 2 бита	Стек муравья. Первые два элемента стека
Dir-End	1 бит	= 1 всегда. Обозначает вершину стека.
Pattern	1 бит	= 0 всегда. Заполнитель до конца флита

В модели используется следующая классификация флитов [2]:

– FP (Full Packet): один флит-пакет.
– EP (End of Packet): последний флит пакета
– BDY (Body): не последний флит

Таким образом, все флиты, кроме последнего, помечены как BDY. Первый флит в пакете может быть определен как первый правильный флит после FP или EP-флита (данная идентификация дает начало механизму маршрутизации).

Таблица 2. Формат BDY-флита F-муравья и B-муравья.

Название	Размер, бит	Описание
IsFAnt	1	0 – B-Ant, 1 – F-Ant
Dir1 … Dir15	по 2 бита	Стек муравья. Максимум 15 элементов стека
Dir-End	1 бит	= 1 всегда. Помещается сразу за последним значением в стеке. Длина стека не постоянна. Обозначает вершину стека в данном флите.
Pattern	1 бит	= 0 всегда. Заполнитель до конца флита

Модель маршрутизатора

На рисунке 5 представлена модель входного и выходного портов маршрутизатора сети на кристалле для алгоритма AntNet.

Входной порт и выходной порт классифицируют пакеты по уровням обслуживания. Для пакетов каждого уровня обслуживания используются отдельные очереди и маршрутная информация. Во входном порту для каждого уровня обслуживания есть параметр CRT (Current Routing Table), хранящий информацию о том, в какой выходной порт необходимо направлять флиты текущего пакета. В выходном порту с каждым уровнем обслуживания сопоставлены два параметра: CSIP (Currently Serviced Input Port) и NBS (Next Buffer State). Первый параметр показывает, какой входной порт в данный момент обслуживается выходным портом. Второй параметр это состояние очереди входного порта того же уровня обслуживания в следующем маршрутизаторе, с которым соединен данный выходной порт.

Параметры CRT и CSIP используются для организации канала передачи флитов одного пакета. Описание значений CRT и CSIP представлены в таблицах 3 и 4, соответственно.

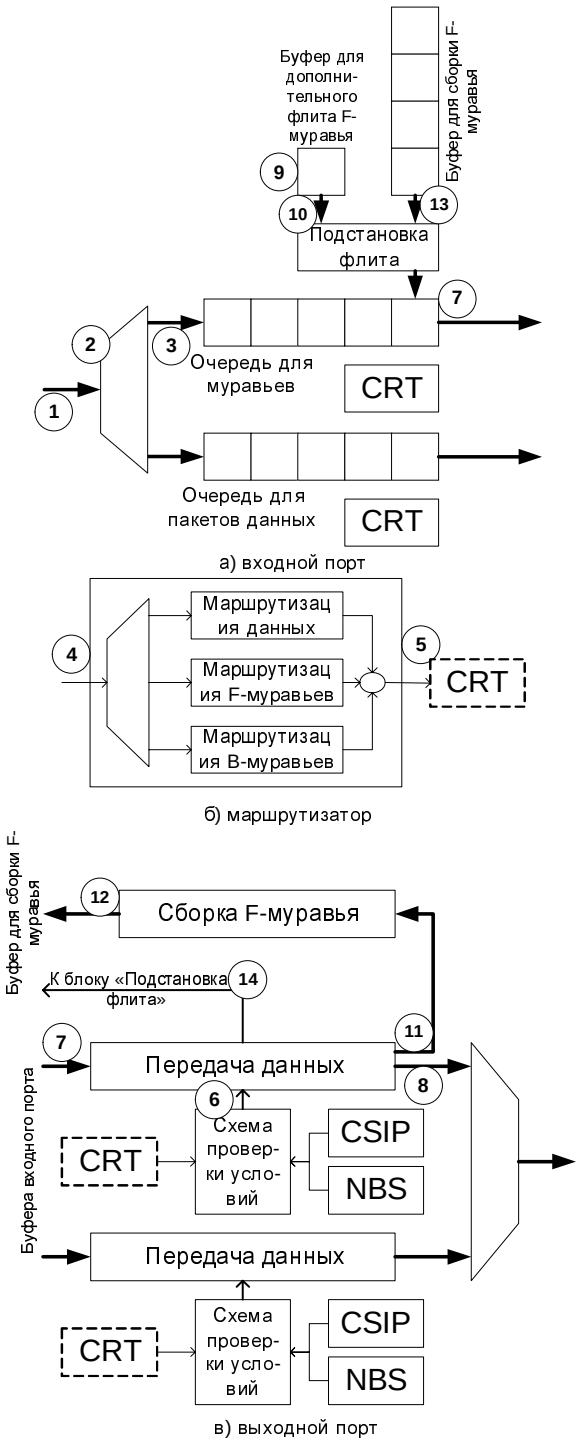

Рисунок 5 – Модель входного и выходного портов маршрутизатора сети на кристалле для алгоритма AntNet

Значение NBS показывает количество свободных ячеек в очереди следующего маршрутизатора, то есть, если NBS=0, то очередь следующего маршрутизатора переполнена.

Алгоритм обработки флитов пакета данных состоит в следующем. Флит попадает во входной порт маршрутизатора (на рис. 5 п. 1). Выполняется классификация флита (на рис. 5 п. 2). Флит ставится в очередь соответствующего уровня обслуживания (на рис. 5 п.3).

Таблица 3. Описание значений CRT

Значение CRT	Описание
CRT >= 0	CRT - номер выходного порта, в который необходимо передавать флиты
CRT = -1	Флит, находящийся в голове входной очереди, еще не маршрутизирован
Дополнительно для AntNet	
CRT = -2	Буфер для сборки F-муравьев содержит собранный F-муравей, переделанный в B-муравья
CRT = -3	Муравей, поступающий на вход должен быть удален (может быть связано с обнулением TTL)

Таблица 4. Описание значений CSIP

Значение CSIP	Описание
CSIP > 0	Тогда CSIP-1 – это номер текущего обслуживаемого входного порта.
CSIP < 0	Тогда |CSIP| - 1 – это номер входного порта, который обслуживался последним.

Как только флит попадает в голову очереди, выполняется его маршрутизация (на рис. 5 п. 4):

1. Если флит – заголовок пакета и CRT = -1, то моделируется задержка на выполнение маршрутизации. После чего выполняется маршрутизация по алгоритму маршрутизации данных. Для пакета выбирается выходной порт. Идентификатор выходного порта записывается в CRT (на рис. 5 п. 5). Если флит – не заголовок пакета, и CRT >= 0, то механизм маршрутизации не запускается.

2. Как только CSIP необходимого выходного порта, соответствующего уровня обслуживания становится отрицательным, запускается механизм выбора обслуживаемого входного порта. Если найден входной порт, ожидающий обслуживания, то CSIP устанавливается равным номеру этого входного порта. CSIP сбрасывается только тогда, когда весь пакет отправлен.

3. После вышеперечисленных действий происходит передача флитов по всем созданным каналам (на рис. 5 п. 6, 7 и 8). Каждый раз при попадании флита в очередь, NBS предыдущего выходного порта уменьшается на 1. При вытаскивании флита из очереди, NBS предыдущего увеличивается на 1. Если NBS = 0, передача флитов блокируется по данному каналу.

Описанный выше алгоритм используется для всех видов пакетов, т.е. является общим механизмом.

Алгоритм обработки флитов муравья включает этапы:

1. В каждом узле с определенной периодичностью создается F-муравей. Момент создания первого муравья – генерируется случайно. Остальные муравьи генерируются через постоянные интервалы времени.

2. На всем пути следования, F-муравей предается с использованием механизмов, сходных с механизмами передачи данных.

3. Каждый раз, когда для F-муравья выбирается CRT направление, обратное выбранному, записывается в тело муравья, в его внутреннюю память. Если эта информация не помещается в текущий, обрабатываемый флит, то она записывается в следующий флит. Если текущий флит был последним, то создается новый и помещается в буфер для дополнительного флита F-муравья (на рис. 5 п. 9). В этом случае последний флит приобретает тип BDY (не последний), и при вытаскивании из очереди на его место помещается флит из буфера для дополнительного флита (на рис. 5 п. 10). NBS не изменяется.

4. Если F-муравей поступил в узел-назначение, то CRT = -2. Это означает, что флиты муравья должны поступать в буфер для сборки F-муравья (на рис. 5 п. 11, 12). Муравей собирается полностью. После сборки он превращается в B-муравья (на рис. 5 п.13), и начинает отправляться в обратном порядке (на рис. 5 п. 14), от последнего флита к головному, по пройденному пути.

5. B-муравей, маршрутизируется также как данные, только маршрутная информация вытаскивается из первого флита (который был последним в F-муравье). Освобожденные от маршрутной информации флиты B-муравья удаляются. По мере приближения к источнику, B-муравей уменьшается в длине.

6. Первые флиты B-муравья содержат исключительно маршрутную информацию. Последний флит содержит значение CS, используемое для обновления таблиц маршрутизации.

Система моделирования сетей на кристалле

Для определения характеристик алгоритмов маршрутизации в сетях на кристалле разработана система моделирования. Эта система выполняется под управлением операционной системы Windows и написана на языке C++ в среде Borland C++ Builder.

Система позволяет задать следующие параметры:

– параметры сети (размер сети, топология, параметры входных портов);
– временные характеристики (время обработки Флитов пакета);
– параметры моделируемых алгоритмов маршрутизации;
– параметры входящего трафика.

Процесс моделирования выполняется последовательно с использованием механизма пошагового продвижения времени. Каждые два шага соответствуют одному синхроимпульсу тактового генератора: фронт импульса и спад импульса, соответственно.

Результатом моделирования являются графики изменения во времени следующих величин, характеризующих поведение сети:

– сгенерированный трафик;
– количество доставленных пакетов;
– нагрузка маршрутизаторов;
– средняя задержка флитов в узлах;
– среднее время доставки флитов.

Все графики можно экспортировать в файлы формата CSV, которые можно загрузить в MS Excel для последующей статистической обработки.

Система позволяет моделировать топологии сети с однородными узлами, такие как «Сетка», «Тор», «Складчатый тор». Разрабатываются модули для построения моделей с неоднородными узлами.

В системе определены базовые классы, на основе которых строятся модели исследуемых алгоритмов маршрутизации, реализованы алгоритмы XY и AntNet для NOC.

Исследование алгоритмов AntNet и XY

С использованием системы моделирования проведены эксперименты с целью изучения поведения алгоритмов AntNet и XY в условиях возникновения перегруженных ячеек на кристалле. Рассматривается сеть с топологией «Тор» размером 8×8. Все входные очереди имеют размер 5 ячеек. Вычисление маршрута для пакетов выполняется за 1 такт. В сети существуют два потока пакетов. Пакеты передаются между парами узлов, находящимися на одном уровне (см. рис. 4).

Первый поток пакетов направлен из узла 26 в узел 29, и запускается в самом начале процесса моделирования. Пакеты поступают в сеть с постоянным периодом 3 такта. Второй поток запускается по истечении 30000 тактов после начала моделирования. Пакеты передаются из узла 27 в узел 28 с тем же периодом в 3 такта.

Графики, показывающие количество пакетов, поступающих в сеть, для алгоритмов XY и AntNet приведены на рисунках 5 и 6 соответственно. Графики, показывающие количество пакетов, доставленных по назначению, для алгоритмов XY и AntNet даны на рисунках 7 и 8 соответственно.

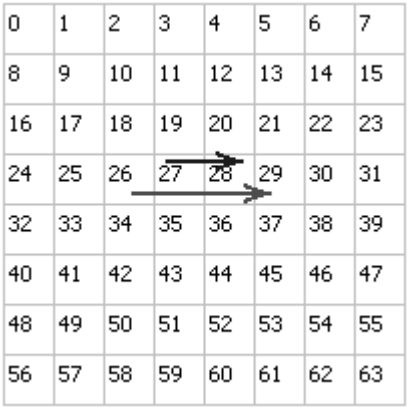

Рисунок 4 – Сеть, используемая в исследовании. Потоки пакетов.

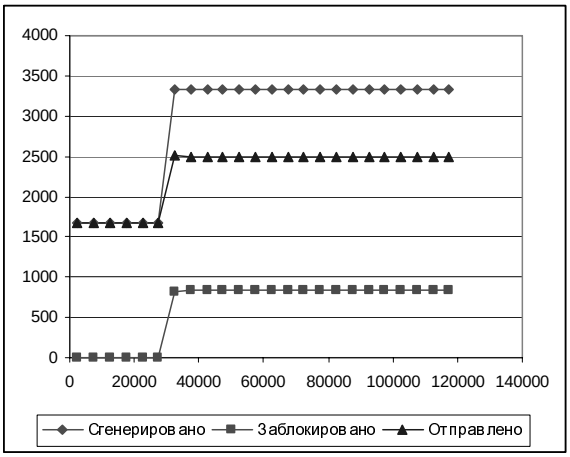

Рисунок 5 – Количество пакетов, поступающих в сеть для алгоритма XY.

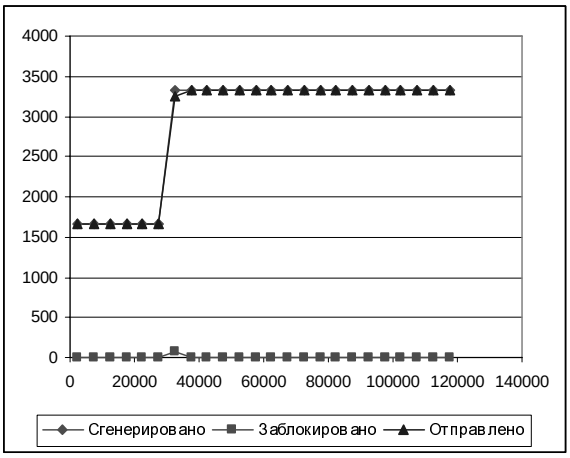

Рисунок 6 – Количество пакетов, поступающих в сеть для алгоритма AntNet.

Экспериментально определено, что в данных условиях время пребывания одного пакета в узле – 2 такта. Если пакеты поступают в узел из обрабатывающего элемента с периодом меньшим, чем через два такта, то входная очередь от обрабатывающего элемента полностью заполняется и отправка новых пакетов в маршрутизатор блокируется. В эксперименте, при отсутствии свободных ячеек в очереди, отправка пакетов отменяется. На рисунках 5 и 6 отмененные пакеты показаны как заблокированные.

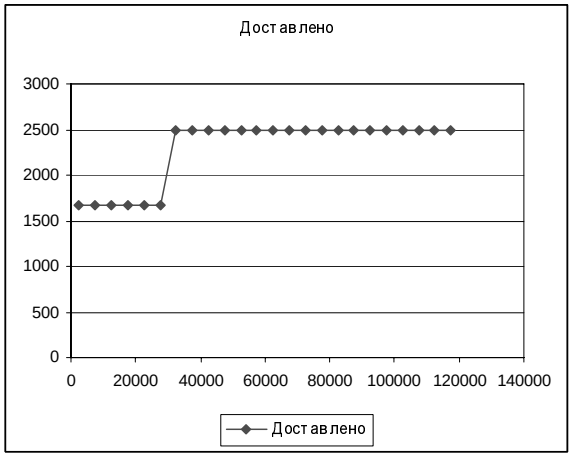

Рисунок 7 – Количество доставленных пакетов для алгоритма XY.

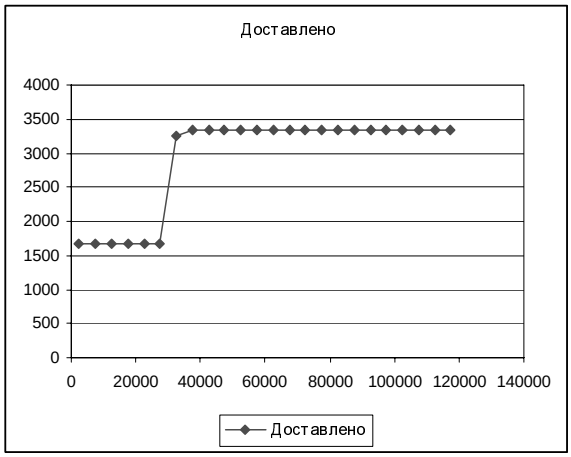

Рисунок 8 – Количество доставленных пакетов для алгоритма AntNet.

Эксперимент построен так, что в момент времени 30000 через узлы 27 и 28 проходит два потока пакетов с периодом в 3 такта. Следовательно, в эти узлы пакеты в среднем поступают с периодом 1,5 такта. Очереди во входных портах переполняются. Маршрутизатор в узле 27 пропускает суммарный поток с периодом не меньше 2 тактов. Блокирование пакетов видно на графике на рисунке 5.

Алгоритм XY является статическим, поэтому маршрут пакетов не меняется. Алгоритм AntNet находит маршрут такой, что пакеты из первого потока обходят перегруженный участок и не блокируются. Это видно на рисунках 6 и 8.

На рисунках 9 и 10 показаны графики времени задержки пактов в узлах.

Рисунок 9 – Задержка флитов в узлах для алгоритма XY.

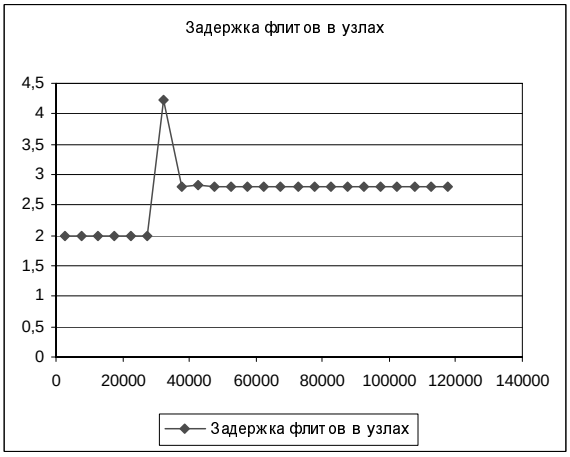

Рисунок 10 – Задержка флитов в узлах для алгоритма AntNet.

В случае с использованием алгоритма XY происходит насыщение пакетами очередей в узле 27 от узла 26 и от PE. Это в свою очередь вызывает насыщение входной очереди в узле 26 от PE. Насыщение очередей приводит к тому, что пакеты начинают дольше находиться в этих очередях, ожидая обработки. Среднее время пребывания пакетов в узлах резко увеличивается. Это видно по графику на рисунке 9.

В случае использования алгоритма AntNet, указанные выше очереди также насыщаются пакетами, но при их переполнении муравьи обнаруживают это состояние, и начинают выбирать другие направления в процессе поиска маршрутов. В некоторый момент времени муравьи обнаруживают лучшие маршруты и перестраивают соответствующим образом таблицы маршрутизации. Часть входной нагрузки перенаправляется на новые маршруты. Нагрузка входных очередей снижается. Описанному переходу от перегрузки к распределению нагрузки по сети соответствует резкий скачек и спад времени пребывания пакетов в узлах сети между отметками времени 30000 и 40000 (рис.10).

Заключение

Разработана модель аппаратной реализации алгоритма AntNet для сети на кристалле. Разработан механизм передачи флитов муравьев, позволяющий заполнять стек муравья информацией о маршруте без предварительной сборки муравья в каждом промежуточном узле по пути его следования.

Представлена система имитационного моделирования сетей на кристалле.

Проведен анализ поведения алгоритмов AntNet и XY в условиях возникновения перегруженных ячеек на кристалле. Получены графики величин, характеризующих поведение сети: сгенерированный трафик; количество доставленных пакетов; нагрузка маршрутизаторов; средняя задержка флитов в узлах; среднее время доставки флитов.

Показано, что при возникновении в сети на кристалле точек скученности алгоритм AntNet перестраивает таблицы маршрутизации и, таким образом, пакеты «обходят» насыщенный участок.

Литература

1. Rantala V., Lehtonen T., Plosila J. Network on Chip Routing Algorithms, TUCS Technical Report № 779, August 2006.
2. Bolotin E., Cidon I., Ginosar R., Kolodny A. QoS architecture and design process for cost effective Network on Chip, CCIT Report №424, Elec. Dept, Technion, May 2003.
3. W.J. Dally and C.L. Seitz. The torus routing chip. Distriduted Computing, 1(3): 187-196, 1986.
4. Daneshtalab M., Sobhani A., Mottaghi M. D., Kusha A.A., Navabi Z., Fatemi O. Ant Colony Based Routing Architecture for Minimizing Hot Spots in NOCs // Proceedings of the 19th annual symposium on Integrated circuits and systems design. 2006. P. 56-61.
5. Hu. J., Marculescu R., DyAD-Smart Routing for Networks-on-Chip, DAC 2004, pp.260-263,204, San Diego, California, USA.
6. Di Caro G., Dorigo M. AntNet: Distributed Stigmergetic Control for Communications Networks: Journal of Artificial Intelligence Research. 1998. №9. P. 317-365.
7. Ладыженский Ю.В., Мирецкая В.А. Исследование муравьиных алгоритмов маршрутизации в компьютерных сетях // Інтернет-Осівта-Наука-2006, п'ята міжнародна конференція ІОН-2006, 10-14 жовтня, 2006. Збірник матеріалів конференції. Том 2. – Вінниця: УНІВЕРСУМ-Вінниця. 2006. С. 375-377.
8. Ладыженский Ю.В., Мирецкая В.А. Муравьиная маршрутизация в сетях на кристаллах. Моделирование и компьютерная графика: Материалы 2-й международной научно - технической конференции , г Донецк , 10-12 октября 2007 г . - Донецк , стр. 182-186.