УДК 004.713, 004.27

А. Шпіцер
Національний університет “Львівська політехніка”,
кафедра спеціалізованих комп’ютерних систем

МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМУ DYXY ШЛЯХОМ СЕГМЕНТАЦІЇ МЕРЕЖІ НА ЗОНИ ЗАВАНТАЖЕНОСТІ

Запропоновано метод підвищення ефективності алгоритму DyXY шляхом сегментації мережі на основі статистичних даних та відносної пропускної характеристики елементів. За цим методом, за наявності кількох вільних альтернативних маршрутів пакет відправляється до сегмента, в якого вища пропускна характеристика. Так зменшується ризик простою пакета через зайнятість транзитного отримувача.

Ключові слова: NoC, топологія, маршрутизація, сегментація, алгоритм DyXY.

A method of increasing the efficiency of the algorithm DyXY by segmenting network based on statistical data and the relative throughput characteristics of elements. This method, in the presence of several free alternative routes, sends package in the segment, in which higher throughput characteristics. This reduces the risk of downtime due to busy transit package recipient.

Key words: NoC topology, routing, segmentation, algorithm DyXY.

Вступ

Розвиток людського суспільства з кожним роком ставить перед ним щораз складніші задачі, які характеризуються великим потоком даних та значною кількістю обчислювальних операцій. Здебільшого ці задачі пов’язані з невпинним процесом глобалізації, який, незважаючи на всі негативні ефекти, зупинити не вдасться. Результатом цих процесів є інтернет як окрема сутність та чимало його ресурсів, таких як ресурси Google, Wikipedia, Skype; система глобальної навігації GPS; метеорологічні дослідження; засоби комунікації та торгівлі; новітні методи освіти, науки та навчання... Подібні приклади можна наводити в усіх сферах життєдіяльності і навіть ті явища, котрі уникнули глобалізації, в найближчому майбутньому будуть змушені так чи інакше взаємодіяти з світом глобальних технологій, якщо не інтегруватися з ним повністю.

Водночас з глобалізацією триває процес мінімізації пристроїв керування, які сьогодні є майже в абсолютній своїй масі цифровими. Як приклад можна навести систему GPS. Велика кількість масивних супутників на орбіті Землі була б металоломом, якби GPS-навігатори не вміщались людині в кишеню. І чи не єдиним застосуванням такої системи можна було б вважати військову галузь. В умовах відносно мирного суспільства створення такої системи було б марнотратством.

Тож розвиток мікро- та нано-, а в найближчому майбутньому і пікотехнологій є зрозумілим та необхідним процесом. Своєю чергою, ці технології дозволили перетворити цифрові пристрої (персонального використання) з персональних обчислювальних станцій на ноутбуки та кишенькові пристрої. Мінімізація пристроїв призвела до їх дискретизації, і машини перетворились на цілі системи, взаємодія елементів яких давала високу продуктивність за низької ресурсозатратності. Окремим класом цих систем стали системи на кристалі (SoC).

Система на кристалі, або система на чипі [1] (від англ System-on-a-chip, або іще SoC чи SOC) – дизайн електронної схеми, яка вміщує функціональні складові цілого пристрою (наприклад, комп'ютера) на одній мікросхемі. Залежно від призначення SoC може оперувати як цифровими сигналами, так і аналоговими, аналого-цифровими, а також частотами радіодіапазону. Типовим застосуванням таких схем є широке різноманіття вбудованих систем.

Якщо не вдається розмістити всі необхідні ланцюги на одному напівпровідниковому кристалі, то використовується схема із декількох кристалів, розміщених на одному корпусі (System

in Package — SiP). SoC вважається вигіднішою конструкцією, оскільки дозволяє збільшувати відсоток придатних схем при виготовленні та спростити конструкцію корпуса.

Типова SoC містить:

• мікроконтролер,
• блок пам'яті,
• джерело опорної частоти,
• таймери, лічильники та ланцюги затримок після вимкнення,
• стандартні інтерфейси для зовнішніх пристроїв: USB, FireWire, Ethernet, UART, SPI,
• входи та виходи цифро-аналогових и аналого-цифрових перетворювачів,
• регулятори напруги та стабілізатори живлення.

Наступний етап розвитку цих технології – це мережі на кристалі (NoC): новий клас пристроїв, коли під рукою знаходяться не один, а тисячі або й мільярди однотипних елементів чи навіть систем, здатних виконувати цілий комплекс простих задач, сукупність розв'язків яких є розв'язком складніших. Такі мережі здатні розпаралелювати обчислення, виконувати кілька ітерацій задачі одночасно. Вони широко застосовуються в різноманітних галузях: обробки сигналів, метеорологічних, космічних та інших дослідженнях, медицині та науці. Здатні забезпечити режим "real time" для будь-яких задач, вони стали передовим напрямком розвитку цифрових технологій.

Існує кілька базових алгоритмів маршрутизації в мережах на кристалі. У цій статті розглянемо алгоритми сімейства XY.

Алгоритм XY [2, 3]

Найпопулярнішим є алгоритм маршрутизації, який запропонували Ванг Ханг і Ліганг Хоу, у якому пакет має заголовок, кінець і дані переміщуються туди, куди вказує адреса в заголовку пакета. Для реалізації двовимірної топології мережі використовується механізм перемикання.

Кожен маршрутизатор, який має координати (x,y), для маршрутизації поточної адреси (Cx, Cy) порівнює адресу призначення роутера (Dx, Dy) пакета. Якщо (Dx > Cx), заголовок рухається на схід, інакше він повертає на захід, поки (Dx, Cx) не стане рівним – це називається горизонтальним вирівнюванням. Тепер (Dy, Cy) піддається стисненню. Якщо буде встановлено, що (Dy < Cy), то заголовок пакета рухається в південному напрямку, інакше – у північному, поки (Dy = Cy).

XY алгоритм відноситься до статичної маршрутизації. При використанні алгоритму XY маршрутизація даних відбувається спочатку в горизонтальному напрямі (по координаті X), а потім у вертикальному (по координаті Y)

Алгоритм XY добре підходить, наприклад, для мереж з регулярною топологією "сітка" (матрична топологія) і "тор".

XY ніколи не приводить до активних тупиків. Однак при такому алгоритмі трафік не розподіляється рівномірно в мережі, а концентрується переважно в її середині. У наслідку виникають істотні затримки при проходженні пакета між віддаленими елементами та перегрівом мікросхеми. Це спричиняє зниження продуктивності мережі, підвищення енергоспоживання, а також підвищення ризику виходу мікросхеми з ладу.

У більшості випадків в NOC (Network-On-Chip) використовується статична маршрутизація. Однак використання цього типу маршрутизації в мережах з нерегулярною топологією небажане.

Алгоритм XYX [4]

Це алгоритм маршрутизації, основна мета якого – забезпечити високу відмовостійкість. Його робота забезпечується створенням надлишкових копій кожного пакета, що відправляється з процесора-джерела і використовує два різні алгоритми, щоб направити оригінал і надлишкові пакети. Оскільки дві копії кожного пакета приходять за призначенням, то з'являється можливість виявити помилкові пакети і замінити їх правильними. Завдяки використанню маршрутів з низьким трафіком для пересилання копій пакетів і мінімізації числа надісланих надлишкових пакетів, XYX

алгоритм дає нижчу продуктивність і більші накладні витрати енергоспоживання порівняно з базовими алгоритмами маршрутизації. Це доводять дослідження, які показують, що алгоритм маршрутизації XYX забезпечує незначну продуктивність і вищу споживану потужність порівняно з тими алгоритмами маршрутизації, що використовуються.

Такі характеристики зумовлені необхідністю в додаткові енергії, щоб надлишкові пакети могли пересуватися по мережі, а швидкодія падає, бо кожен вузол, що обробляє пакети, має перевірити оригінал і копію на наявність помилок, і якщо їх виявлено, то виправити їх.

Маршрутизатори можуть відрізнити оригінальний пакет від надлишкових за допомогою біта індикації на початку пакета. Оригінальні пакети відправляються за XY маршрутом, тоді як надлишкові пакети відправляються за YX маршрутом, який є протилежним до XY. У YX маршрутизації пакет відправляється за Y напрямом на першому етапі, а коли пакет досягне стовпця, де знаходиться вузол призначення, він направляється за X кординатою, щоб досягти вузла призначення. За рахунок цього трафік майже рівномірно розподіляється по мережі, тому всі канали роблять однаковий внесок у розподіл трафіка.

Для того, щоб мінімізувати накладні витрати, зменшують кількість надлишкових пакетів до одного на кожен оригінальний.

Псевдоадаптивний XY [5]

Псевдоадаптивний XY алгоритм являє собою продовження класичного XY. Навантаження в центрі мережі у звичайному XY методі маршрутизації набагато більше, ніж середньозагальне. Основна мета цього алгоритму маршрутизації – це розподіл навантаження в мережі. Основною перевагою розподілу навантаження є збалансована температура мікросхеми. Цей алгоритм має два режими: детермінований і адаптивний. Статус сусідів кожного комутатора використовується для того, щоб вибрати правильний режим. Пакети направляються згідно з XY алгоритмом (детермінований режим), коли навантаження в мережі є низьким. Коли навантаження високі, пакети проходитимуть через менш завантажений маршрут (адаптивний режим).

Детермінований режим дає меншу затримку під час передавання пакетів, але він ефективний при низькому трафіку, а адаптивний дає більшу затримку під час передавання, але за високого трафіка в мережі він працює ефективніше, оскільки обирає найоптимальніший шлях для пакета і забезпечує ефективне використання ресурсів.

Комутатори наділені певним “інтелектом”, щоб розрізняти статус своїх сусідів. Отримуючи пакет за допомогою перемикача, він намагається скерувати пакет за XY маршрутизацією до одного із своїх сусідів, але якщо навантаження сусіднього комутатора більше за встановлений поріг, то обирає сусіда з меншим навантаженням. Цей спосіб реалізований в комутаторі так: кожен його порт, крім локальних, має два біти, які визначають статус руху в цьому порті (їх ще називають біти квантового навантаження). Так що кожен порт має чотири квантові значення, які відправляються північному, південному, західному та східному сусідам. Під час маршрутизації використовується середнє число пакетів, що пройшли за кілька тактів для розрахунку навантаження в кожному комутаторі.

У комутаторі реалізовано пріоритети, за якими пакети надходять до нього. Наприклад, коли приходять два пакети одночасно, встановлюється фіксований пріоритет. Пакети, що приходять з півночі мають найвищий пріоритет, потім ті, що прийшли з півдня, потім – зі сходу і, нарешті вхідні пакети, що надійшли із заходу отримують найнижчий пріоритет. Наприклад, ми хочемо передати пакет із північного заходу до центру мережі. Спочатку обираємо XY алгоритм, щоб відправити пакети до центру. Якщо немає високого навантаження, то продовжуємо рух, в іншому випадку перевіряємо статус навколишніх комутаторів. Якщо всі навколишні комутатори з низьким навантаженням, то ми залишаємо XY маршрут у іншому випадку застосовуємо адаптивний XY алгоритм для обрахунку нового маршруту. Завдяки цьому не треба очікувати звільнення якогось одного комутатора. Для пересування пакетів за цим алгоритмом збільшують розмір буфера і добавляють додаткову логіку керування буфером, щоб уникати тупиків, які є основною проблемою маршрутизації.

TXY алгоритм [6, 7]

Алгоритм перемикання (Toggle XY (TXY)) покращує балансування навантаження під час маршрутизації завдяки тому, що одна половина пакетів йде XY маршрутом, а інша половина – YX маршрутом. Кожен заголовок пакета містить біт, що вказує, який маршрут у пакета: XY або YX. Кожен CNI (connection network interface) здійснює перемикання між XY та YX пакетами. Ця схема дозволяє уникнути тупиків за допомогою двох віртуальних каналів на маршрутизаторі: один для XY шляху, а інший для YX з будь-якою схемою планування між віртуальними каналами.

На CNI покладаються такі функції: фізичний мережний інтерфейс, буферизація, перевпорядкування, фрагментація/складання (застосування конкретних блоків в пакетах) і підтримка маршрутизації.

Деякі особливості алгоритму:

• Оптимальний для симетричного трафіка;

• Вдалий баланс навантаження;

• Розщеплені маршрути;

• Не враховується схема руху.

Розподіляючи трафік між XY та YX маршрутами, в TXY не завжди вдається досягти оптимального балансу завантаженості. В деяких випадках, посилаючи різні частини трафіку, кожен маршрут може досягти кращого розподілення навантаження і ,відповідно, знизити вимоги до продуктивності.

DyXY алгоритм [8]

Це новий алгоритм, запропонований для NoC, щоб забезпечити адаптивну маршрутизацію та роботу без статичних і динамічних (взаємоблокування та активний тупик) тупиків одночасно. Для підтримки цього алгоритму була розроблена нова архітектура. Аналітичні моделі, що основуються на теорії масового обслуговування, також були адаптовані для DyXY маршрутизації в двовимірної мережі. DyXY маршрутизацією досягають вищої продуктивності порівняно із стандартною XY і odd-even маршрутизацією.

Адаптивність полягає в прийнятті рішення з маршрутизації, враховуючи результати моніторингу стану перевантажень безпосередніх сусідів і можливістю виникнення статичних і динамічних тупиків, що спричинені особливістю алгоритму XY надсилати одним з найкоротших шляхів між джерелом і приймачем. Якщо існує декілька найкоротших шляхів, роутери допомагають пакету вибрати один з них, враховуючи стан перевантаженості мережі. Детальніше алгоритм можна розглянути наступним чином:

1. Прочитати значення вхідних пакетів.
2. Порівняти адреси призначення і поточного маршрутизатора

• Якщо пунктом призначення є локальне ядро поточного маршрутизатора, відправити пакет в локальне ядро.

• Інакше:

Якщо пункт призначення має ту саму x (або y) адресу поточного роутера, відправити пакет сусідньому роутеру на y-вісь (x-вісь відповідно) до приймача.

Інакше перевірити значення навантаження безпосередніх сусідів поточного маршрутизатора, які є на шляху до місця призначення і відіслати пакет сусіду з найменшим навантаженням.

Суттєвим недоліком даного алгоритму є незначний аналіз мережі, який обмежується безпосередніми сусідами.

Суть методу

Необхідно забезпечити алгоритму DyXY можливість проаналізувати сегмент мережі, більший, ніж безсердні сусіди. Є два можливі підходи до розв’язання цієї задачі:

1) реалізація супервізора (віртуальний пристрій, що контролює роботу всієї мережі та виконує інші глобальні сервісні задачі), який би мав доступ до всіх елементів та міг на основі їх стану спрогнозувати найоптимальніший маршрут;

2) забезпечити можливість алгоритму отримувати дані про віддалені елементи через безпосередніх сусідів.

Реалізація першого методу, а саме створення супервізора, має кілька суттєвих недоліків:

1) супервізор потребує частини ресурсів мережі;

2) алгоритм роботи супервізора потребує додаткових, суттєвих часових затрат на кожному кроці пакета.

Тому детальніше розглянемо другий метод.

Оскільки кожен елемент-відправник немає прямих зв'язків з опосередкованими елементами (згідно із матричною топологією, яка розглядається), то потрібно модифікувати елементи, надавши їм інформацію про віддалені ділянки мережі.

Модифікація елемента полягає у виділенні частинки пам'яті під сервісну інформацію, а саме тип та номер сегмента, до якого він належить.

Наступним кроком має бути сегментація мережі. Для цього можна використати будь-який із відомих алгоритмів сегментації зображення чи трасування, попередньо модифікувавши його. У цьому дослідженні використовується відповідно модифікований алгоритм "Краскала" [9] через простоту його реалізації.

У цій задачі виникають питання, вирішення яких залежить від поставлених цілей:

1) що виконуватиме програму сегментації;

2) як часто проводити сегментацію;

3) які критерії завантаженості виставляти.

Ці питання не стосуються теми даної статті, тому розглядаються в загальних рисах. Виконувати сегментацію, як і будь-яку іншу задачу в мережах на кристалі, можна або засобами самої мережі, або за допомогою супервізора.

При сегментації мережі її власними засобами процес є триваліший, оскільки потрібно запустити один чи кілька пакетів у мережу. Ці пакети повинні пройти всі елементи та зібрати інформацію про їх завантаженість, після чого ця інформація обробляється та на її основі проводиться сегментація. Коли сегменти визначені, в мережу потрібно відправити наступні пакети, які проставлять елементам тип та номер сегмента, до якого він належить. Якщо є можливість зупинити роботу мережі на час сегментацію, то ця процедура є нетривалою, в іншому випадку вона може зайняти доволі багато часу залежно від завантаженості мережі.

За допомогою супервізора, який має безпосередній доступ до всіх елементів в будь-який момент часу, сегментацію можна провести значно швидше. Недоліком такого підходу є затрата ресурсів на реалізацію супервізора.

Для вибору підходу, за яким проводити сегментацію, варто керуватись таким принципом: якщо супервізор передбачений для інших цілей, то використовувати його в іншому випадку сегментувати засобами мережі. Проте, як згадувалось раніше, остаточне рішення необхідно приймати залежно від поставленої задачі.

Незалежно від вибраного підходу, частота сегментації також загалом визначається з двох міркувань:

1) чи є в даний момент вільні ресурси (супервізора чи мережі) для сегментації;

2) чи є необхідність сегментації (залежить від динамічності мережі, яка визначає, як швидко змінюється завантаженість елементів та чи змінюється вона в принципі).

Щодо критеріїв сегментації, то вони є доволі суб'єктивними і залежать також від стану та динамічності мережі.

Опис алгоритму

Як і алгоритм DyXY, представлений алгоритм розроблений для матричної топології та проводить маршрутизації найкоротшим маршрутом. Представлений алгоритм розглядає ситуацію, коли наявні кілька альтернативних маршрутів (при матричній топології альтернативних може бути два шляхи).

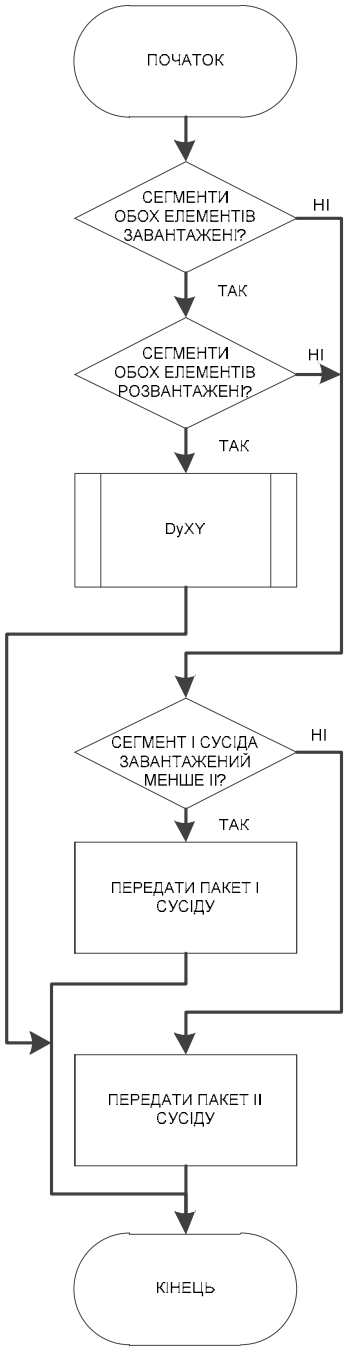

Рис. 1. Алгоритм SDyXY

Експериментальні дослідження

Тестували алгоритм засобами програмного забезпечення MDTNoC [10].

Параметри тестування:

1) розмір мережі змінюється від 6x6 до 99x99 елементів;
2) час, необхідний для обробки пакета елементом, отримувач не враховує;
3) пакет відправляється з елемента 0:0 до елемента N:N, де N – розмірність мережі;
4) досліджується рух лише одного пакета;
5) завантаженість елементів встановлюється в межах: від 0 % до 100 %, від 0 % до 20 %, від 20 % до 80 %, від 80 % до 100 %;
6) алгоритм SDyXY порівнюється з алгоритмом DyXY;
7) результатом є кількість тактових імпульсів, затрачених на проходження пакета від відправника до отримувача.

Було проведено два експерименти та отримано такі результати:

Завантаженість 0–100 % (рис. 2).

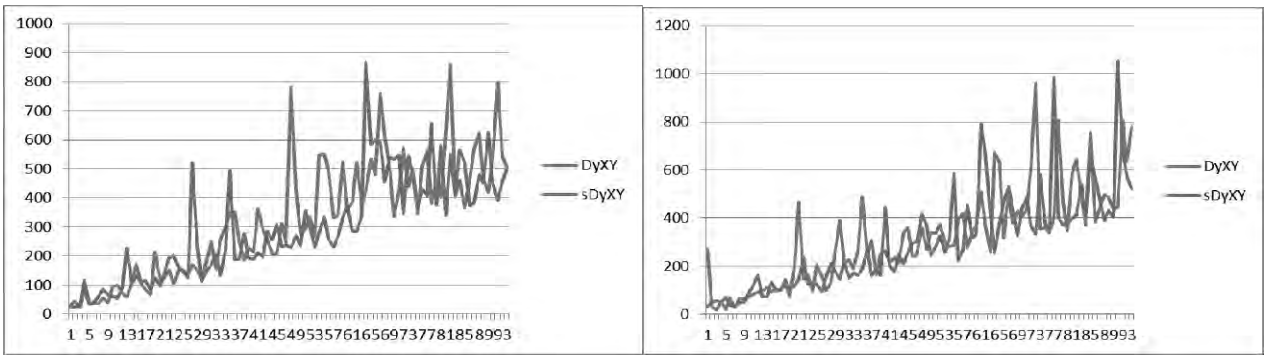

Рис. 2. Завантаженість 0–100 %

Завантаженість 0–20 % (рис. 3).

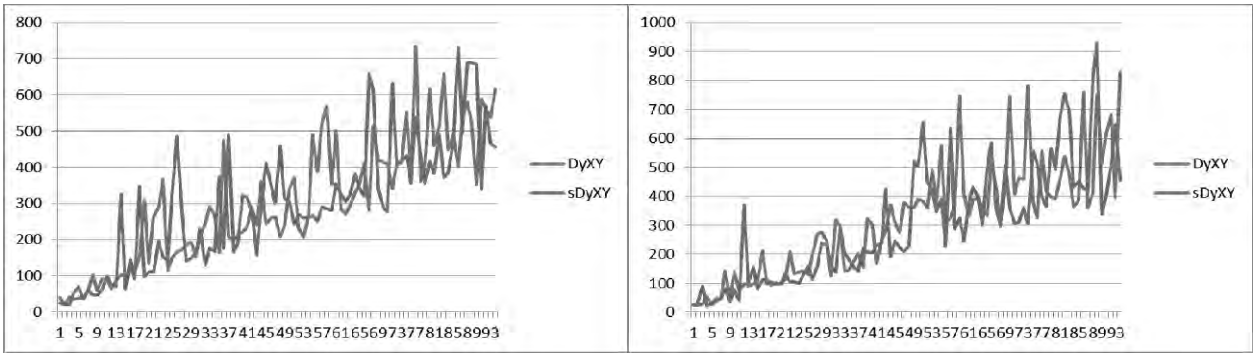

Рис. 3. Завантаженість 0–20 %

Завантаженість 20–80 % (рис. 4).

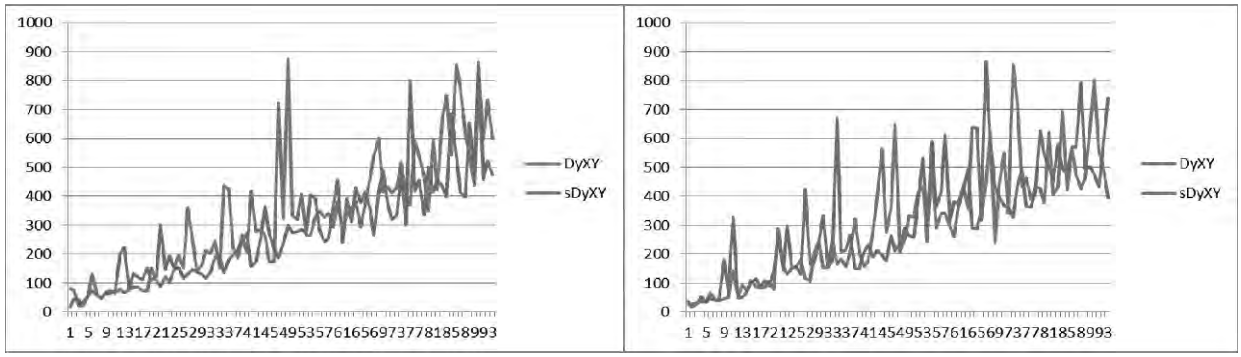

Рис. 4. Завантаженість 20–80 %

Завантаженість 80–100 % (рис. 5).

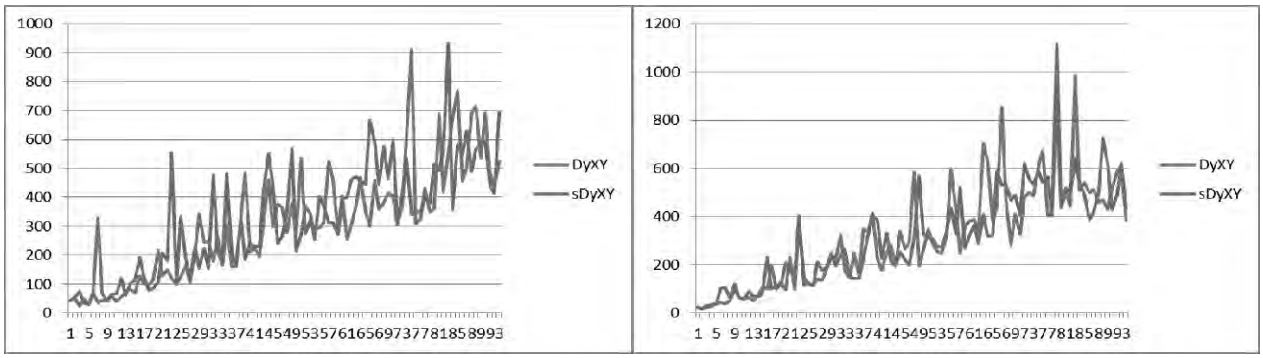

Рис. 5. Завантаженість 80–100 %

Середня арифметична ефективність SDyXY порівняно з DyXY:

I експеримент

- від 0 % до 100 % – 8 %;
- від 0 % до 20 % – 2 %;
- від 20 % до 80 % – 13 %;
- від 80 % до 100 % – -4 %.

II експеримент

- від 0 % до 100 % – 7 %;
- від 0 % до 20 % – 4 %;
- від 20 % до 80 % – -1 %;
- від 80 % до 100 % – 0 %.

Висновки

Аналіз отриманих результатів показує стабільну ефективність алгоритму SDyXY в межах 5–10 % порівняно з DyXY. У разі слабко чи сильно завантажених мереж результати тестування знаходяться в межах статистичної похибки моделювання, тому можна вважати, що обидва алгоритми проявили себе однаково. При середньозавантажених мережах результат залежить від конкретної ситуації, але можна стверджувати, що SDyXY не програє DyXY, а в деяких випадках може дати ефективність до 20 %.

Розробляючи мережу на кристалі, рішення про використання SDyXY приймають на основі отриманих статистичних даних, проте не варто забувати про час та ресурси, необхідні для сегментації мережі, без якої алгоритм SDyXY не зможе працювати.

До позитивних аспектів алгоритму SDyXY належать рівномірне завантаження мережі, чим не володіє DyXY. Цей аспект знижує нагрівання окремих ділянок мережі та енергоспоживання.

1. Badawy W. System-on-chip for real-time applications / W. Badawy, G. Jullien // The Springer International Series in Engineering and Computer Science. – Kluwer, 2003. – P.465 2. QoS architecture and design process for cost effective Network on Chip / E.Bolotin, I.Cidon, R.Ginosar, A. Kolodny // Journal of Systems Architecture, special issue on Networks on Chip. – Israel: Technion-Israel Institute of Technology, 2003. – № 424 3. Ant Colony Based Routing Architecture for Minimizing Hot Spots in NOCs / M.Daneshtalab, A.Sobhani, M.Mottaghi, A.Kusha and other // Proceedings of the 19th annual symposium on Integrated circuits and systems design. – NY: ACM, 2006. – P.56–61 4. Patooghy A. XYX: A Power & Performance Efficient Fault-Tolerant Routing Algorithm for Network on Chip / A.Patooghy, Miremadi S. // 17th Euromicro International Conference. –Tehran: Sharif Univ. of Technol., 2009. – P.455 5. Evaluation of Pseudo Adaptive XY Routing Using an Object Oriented Model for NOC / M.Dehyadgari, M.Nickray, A.Afzali-kusha, Z.Navabi // 17th Euromicro International Conference. – Iran: Tehran Univ., 2005. – P.361 6. A NoC Simulation & Verification Platform based on SystemC / S. Chai, C.Wu, Y. Li, Z. Yang // International Conference on Computer Science and Software Engineering. – China: Univ. of Electron. Sci. & Technol., 2008. – P.522. 7. NoC-Based FPGA: Architecture and Routing / R.Gindin, I.Cidon, I.Keidar // First International Symposium. – Haifa: Technion-Israel Inst. of Technol., 2007. – P.334 8. DyXY – A Proximity Congestion-Aware Deadlock-Free Dynamic Routing Method for Network on Chip / M.Li, Q.Zeng, W.Jone // Design Automation Conference. – OH: Dept. of Electr. & Comput. Eng., Cincinnati Univ., 2006. – P.1126 9. Шпіцер А.С. Стислий огляд методів сегментації та адаптація одного з них для рішення задачі сегментації мережі за критерієм завантаженості елементів // Науково-технічний журнал "Радіоелектронні і комп'ютерні системи". – Харків: НАУ ХАІ. – 2012. – № 5(57). – С.95–99 10. Шпіцер А.С. Інструментальний засіб сегментації мережі на кристалі // Материалы XVI международного молодежного форума "Радиоелектроника и молодежь в XXI веке". – Харків, 2012. – С.15–16.