УДК 004.715

Короткий Є.В. (ФЕЛ НТУУ „КПІ”)
д.т.н. Лисенко О.М. (ФЕЛ НТУУ „КПІ”)

ВИСОКОПРОДУКТИВНА МОДЕЛЬ МАРШРУТИЗАТОРА ДЛЯ МЕРЕЖІ-НА-КРИСТАЛІ З АГРЕГАЦІЄЮ КАНАЛІВ

Запропоновано та досліджено високопродуктивну модель маршрутизатора для мережі-на-кристалі (МнК) з агрегацією каналів (АК), описану за допомогою SystemC. Результати симуляції в ModelSim свідчать, що використання розробленої моделі замість її деталізованого RTL аналогу на SystemVerilog дозволяє на порядок зменшити тривалість моделювання і скоротити в 121 раз обсяг використаної пам'яті. При цьому для моделі на SystemC помилка передбачення порогу насичення МнК не перевищує 6.1%.

Короткий Е.В., Лысенко А.Н. Высокопроизводительная модель маршрутизатора для сети-на-кристалле с агрегацией каналов. *Предложена и исследована высокопроизводительная модель маршрутизатора для сети-на-кристалле (СтнК) с агрегацией каналов (АК), описанная при помощи SystemC. Результаты симуляции в ModelSim свидетельствуют, что использование разработанной модели вместо ее детализированного RTL аналога на SystemVerilog позволяет на порядок сократить продолжительность моделирования и уменьшить в 121 раз объем используемой памяти. При этом для модели на SystemC ошибка предсказания порога насыщения СтнК не превышает 6.1%.*

I.Korotkyi, O.Lysenko A high performance model of router for network-on-chip with link aggregation. *The high performance SystemC model of the router for network-on-chip (NoC) with link aggregation (LAG) is proposed and investigated. The results of the simulation in ModelSim show that employing of the proposed model instead of its detailed System Verilog counterpart allows to reduce modeling time by an order and decrease the memory usage in 121 times. At the same time, the error of prediction of saturation threshold for the developed SystemC model does not exceed 6.1%.*

Ключові слова: *мережа-на-кристалі, агрегація каналів, модель, маршрутизатор, SystemC*

Вступ

Однією з характерних тенденцій, притаманних сучасним системам-на-кристалі, є постійне зростання числа обчислювальних модулів при одночасному підвищенні вимог до кількості та ємності міжмодульних з’єднань [1]. Внаслідок ускладнення внутрішньої структури систем-на-кристалі такі способи організації підсистеми зв’язку як „загальна шина” і „повнозв’язна архітектура” втрачають ефективність [2, 3]. Для вирішення проблеми обміну даними всередині надвеликих інтегральних мікросхем (ІМС) запропоновано використовувати концепцію мереж-на-кристалі (МнК) [4]. Такий підхід має переваги масштабованості, паралелізму і високої тактової частоти [1]. Уявлення про МнК можна отримати з оглядових робіт [1, 5 – 8].

Для оцінки ефективності функціонування МнК важливими характеристиками є транспортна затримка і пропускна здатність. Під транспортною затримкою розуміють інтервал часу між створенням пакета і його прийомом в пункті призначення, включаючи тривалість очікування в буфері передавача. Пропускна здатність характеризує максимальну кількість даних, що передаються в мережі за одиницю часу і обмежується порогом насичення – значенням прикладеного навантаження, при якому транспортна затримка МнК зростає в десятки разів внаслідок конфліктів за використання ресурсів.

На відміну від макромереж до конструкції МнК висуваються більш суворі вимоги на кількість апаратурних ресурсів, необхідних для її реалізації, тому для МнК характерне застосування коротких черг на входах маршрутизаторів, що робить недоцільною буферизацію всього пакету перед відправкою у порт призначення [1]. Замість цього використовується технологія „wormhole” передачі даних, коли пакет розбивається на атомарні одиниці управління потоком (фліти), що передаються безперервно один за одним [9]. Фліти (flit – flow control unit) просуваються по мірі можливості, не чекаючи приходу послідовників. Це забезпечує низькі вимоги до обсягу буферного простору. При такому підході різко зростає ймовірність блокування голови колони (БГК). БГК виникає, коли два (або більше) пакети з різних вхідних черг потребують передачі через один і той же вихідний порт. В такому випадку буде задоволений лише один запит на з’єднання. Ті входи, які програли процедуру арбітражу, повинні чекати, доки необхідний вихідний порт звільниться,

створюючи передумови для виникнення БГК в сусідніх маршрутизаторах. Оскільки при wormhole передачі даних один пакет може займати одночасно кілька маршрутизаторів, це збільшує вірогідність БГК. Результати [10] свідчать, що внаслідок БГК пропускна здатність wormhole мережі обмежується 50 % її ємності.

Використання віртуальних каналів (ВК) є ефективним методом зменшення ймовірності виникнення БГК [10]. На жаль такий підхід вирішує проблему БГК лише частково, оскільки в разі застосування ВК віртуальні потоки мультиплексуються через єдине фізичне з’єднання між сусідніми маршрутизаторами, що накладає обмеження на значення порога насичення мережі [11].

В [11, 12] запропоновані структурні і апаратурні рішення для реалізації агрегації каналів (АК) в wormhole МнК, що дозволяють усунути БГК і значно (на 300 %) підвищити поріг насичення у порівнянні з класичною wormhole МнК. Зазначений результат досягається за рахунок використання кількох фізичних зв’язків для з’єднання сусідніх маршрутизаторів. Недоліком АК є значний обсяг апаратурних ресурсів, необхідних для реалізації МнК у порівнянні з класичною wormhole архітектурою (без ВК). Одним зі способів зменшення апаратурних витрат є проведення параметричної оптимізації кількості зв’язків всередині агрегованих логічних з’єднань. Виконання оптимізації такого роду вимагає створення швидкодіючої моделі об’єкту дослідження, **що і визначає мету роботи**.

В пропонованій статті розглянуто способи зменшення тривалості симуляції SystemC моделей ІМС та переваги використання цієї мови для виконання параметричної оптимізації цифрових систем. Представлено та досліджено алгоритм функціонування поведінкової SystemC моделі маршрутизатора для МнК з АК. У порівнянні з RTL моделлю, описаною з використанням System Verilog [12], запропоноване рішення дозволяє на порядок зменшити тривалість симуляції і скоротити обсяг використаної пам’яті в 121 раз.

Робота має наступну структуру. Розділ 1 присвячений аналізу переваг та недоліків методу передачі даних в МнК з використанням ВК. У розділі 2 розглядається архітектура і принцип функціонування маршрутизатора для МнК з АК. В розділі 3 проаналізовано способи підвищення продуктивності симуляції моделей ІМС та обгрунтовано використання мови SystemC з цією метою. В наступному розділі представлено алгоритм функціонування високопродуктивної моделі маршрутизатора для МнК з АК. В розділі 5 за допомогою комп’ютерного моделювання досліджено залежність порогу насичення МнК від числа з’єднань в агрегованих каналах для двох моделей – швидкодіючої на SystemC і детальної (RTL) на System Verilog. Там же виконано порівняльний аналіз точності і продуктивності досліджуваних моделей. Останній розділ містить висновки та інформацію про подальші дослідження у даному напрямку.

1. Переваги та недоліки wormhole МнК з ВК

У попередньому розділі розглянуто проблему БГК в wormhole МнК. Для зниження вірогідності БГК William Dally в [10] запропонував архітектуру маршрутизатора з ВК. Відповідно до [10] суть методу передачі даних з використанням ВК у wormhole МнК полягає в наступному. Кожному із входів маршрутизатора ставиться у відповідність кілька буферів, кожен з яких відповідає одному ВК. У випадку блокування певного ВК для передачі пакетів можуть бути використані інші, вільні ВК (якщо такі є в наявності). Таким чином, зменшується вірогідність блокування з’єднань між маршрутизаторами і підвищується поріг насичення мережі.

Аналіз описаних в [13 – 16] маршрутизаторів, а також результати власних досліджень [17] дозволяють авторам зробити висновок, що використання ВК у wormhole МнК приводить до збільшення порогу насичення не більше, ніж на 20% від ємності мережі (у порівнянні з wormhole МнК без ВК, враховуючи постійну кількість буферної пам’яті на порт маршрутизатора для обох випадків). Існує границя, при перевищенні якої подальше збільшення кількості ВК майже не впливає на поріг насичення мережі. В [14] показано, що при збільшенні кількості ВК з 2-х до 4-х, поріг насичення МнК зростає з 23% ємності мережі

до 25 %. Іншими словами, збільшення числа ВК на 100 % призводить до зростання порогу насичення лише на 2 %. Подібна залежність присутня також в роботах [16, 17]. Враховуючи, що між числом ВК і кількістю апаратурних ресурсів, необхідних для реалізації маршрутизатора, існує лінійна залежність з коефіцієнтом пропорційності близьким до одиниці [12, 14], з наведених цифр можна зробити висновок про відсутність масштабованості методу ВК. З точки зору авторів описана проблема пояснюється тим, що пропускна здатність фізичного з’єднання між маршрутизаторами розділяється поміж ВК, що використовують його, а це в свою чергу призводить до збільшення періоду передачі флітів через відповідні логічні зв’язки.

З цього можна зробити висновок, що з ростом числа ВК єдине фізичне з’єднання між сусідніми маршрутизаторами стає вузьким місцем системи і обмежує подальше зростання порогу насичення МнК та її пропускної здатності.

2. Маршрутизатор для МнК з АК

Як показано вище, перевагою і недоліком методу ВК є поділ фізичної лінії зв’язку на логічні з'єднання. Застосувавши винахідницький метод інверсії [18], авторами запропонована зворотна концепція – об’єднання кількох фізичних зв’язків в агрегований логічний канал (транк). Іншими словами, для з’єднання топологічно сусідніх маршрутизаторів пропонується використовувати кілька просторово розділених фізичних каналів (ФК), по кожному з яких передаються фліти різних пакетів. Отже, через агреговане логічне з’єднання можна передавати одночасно N пакетів даних, де N – кількість ФК у транку. Запропонований підхід дозволяє знизити ймовірність БГК та збільшити поріг насичення і пропускну здатність МнК. З рис.1 видно, як агрегація трьох ФК усуває БГК на західному вході маршрутизатора M_{11}. У випадку, коли відомий просторовий розподіл потоків даних в мережі, існує можливість підібрати число ФК для кожного транка таким чином, щоб повністю виключити БГК.

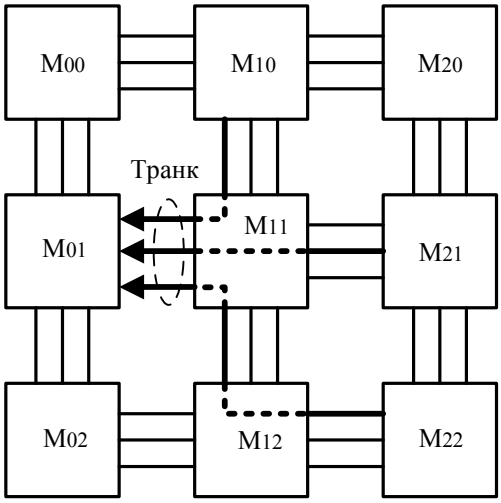

Рис.1. Усунення БГК в МнК з агрегацією 3-х фізичних каналів:
М – маршрутизатор

Аналіз літературних джерел показав, що агрегація каналів (АК) вже використовується в макро мережах, а документи [19, 20] стандартизують АК для Ethernet. Оскільки конструкторські рішення, прийняті в Ethernet, зважаючи на свою складність та ресурсоємність непридатні для використання в МнК, стає актуальним завдання створення простого та ресурсоефективного механізму реалізації АК в інтегральних мережах. Одне з можливих розв’язків такого завдання розглядається далі.

На рис.2 наведена структурна схема маршрутизатора для wormhole МнК з АК. Замість терміну „порт” для опису входів-виходів такого пристрою використовується поняття „транк”, визначення якого розглядалося вище. Запропоноване рішення включає N вхідних і N вихідних транків, кожен з яких містить М ФК. Сполучення ФК на входах маршрутизатора

з ФК на його виходах здійснюється за допомогою комутатора розмірністю NMxNM. Модуль керування містить блоки маршрутизації, керування потоком, виділення ФК, а також логіку, що реалізує алгоритм роботи маршрутизатора. Для кожного вхідного ФК прибуваючі фліти буферизуються в виділеній черзі, оскільки у разі, коли кількість ФК менше кількості потоків даних, що протікають через транк, негайне обслуговування пакету після його прибуття у вхідний ФК не гарантовано.

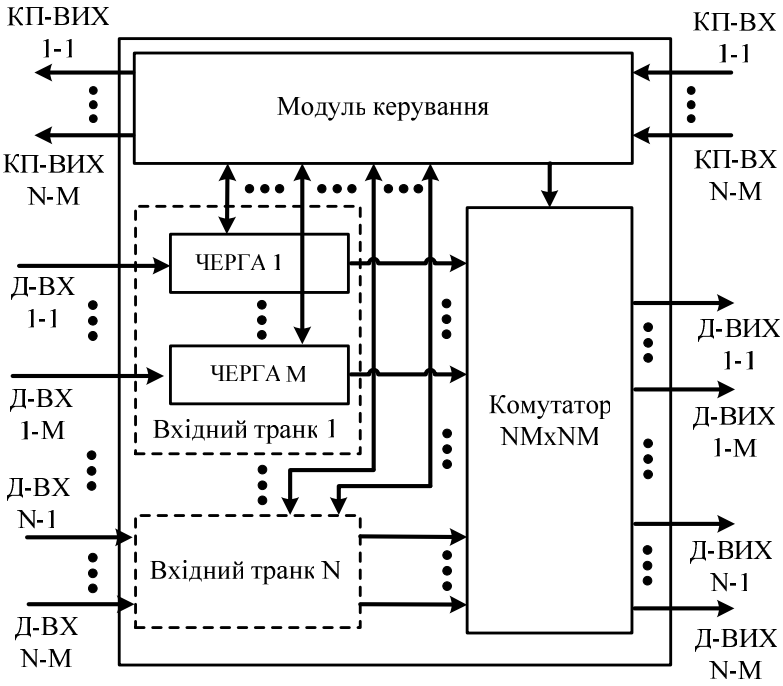

Рис.2. Структурна схема маршрутизатора для МнК з АК:
КП – керування потоком; Д-ВХ – вхід даних; Д-ВИХ – вихід даних

Затримка передачі такого маршрутизатора складає два цикли тактової частоти. На першому етапі відбувається маршрутизація і виділення ФК, на другому – комутація, просування фліта через маршрутизатор та його вилучення з відповідної вхідної черги. Гарантована доставка даних забезпечується застосуванням механізму керування потоком на основі „кредитів” [21]. Розглянемо згадані операції більш докладно.

Маршрутизація

Для зменшення транспортної затримки в запропонованому рішенні використовується завчасна (look ahead) маршрутизація, коли напрямок передачі фліта для поточного маршрутизатора визначається у попередньому вузлі мережі [21]. Такий підхід дозволяє розпочати встановлення з’єднання одразу ж після приходу першого з флітів пакету, одночасно виконуючи обчислення транка призначення для наступного маршрутизатора. Під транком призначення мається на увазі агрегований канал, у який необхідно перенаправляти всі фліти даного пакету. Таким чином, маршрутна інформація для поточного маршрутизатора міститься у спеціальному службовому полі першого з флітів пакету і доступна на початку циклу тактової частоти.

Виділення ФК і комутація

Для просування даних через маршрутизатор необхідно сполучити ФК на вході з одним із ФК на виході. У даному рішенні з'єднання між входом і виходом встановлюється під час приходу першого фліта і розривається після відправки останньої частини пакету. В момент роз’єднання відповідний ФК на виході стає вільним і знову доступним для призначення одному із входів. Після приходу першого фліта пакету кожен із вхідних ФК робить спробу отримати вільний ФК у транку призначення шляхом встановлення запиту на відповідному вході блоку виділення ФК (рис.3).

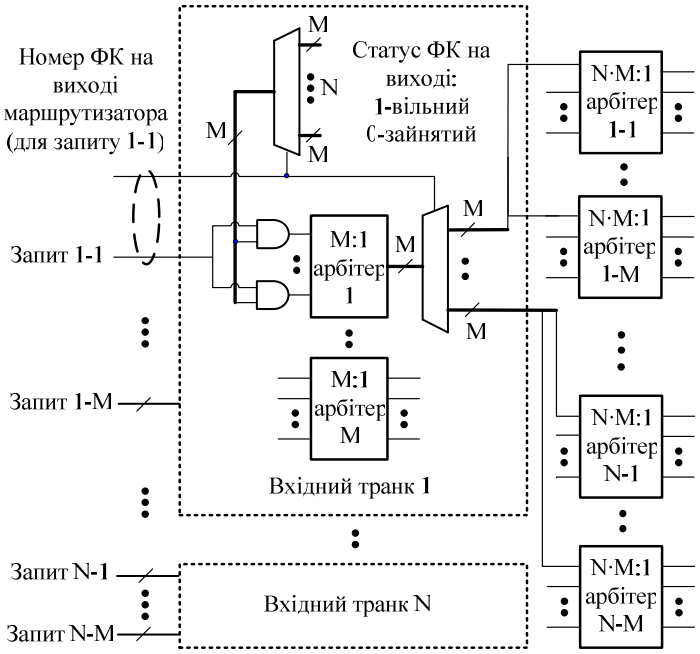

Рис.3. Структурна схема блоку виділення ФК

У випадку успішного завершення цієї операції згаданий запит знімається і між відповідними ФК на вході і виході маршрутизатора встановлюється з’єднання за допомогою комутатора. Число входів блоку виділення ФК дорівнює кількості вхідних ФК маршрутизатора і кожен такий вхід складається з однобітної лінії запиту та шини, що містить номер транка призначення. Як видно з рис.3, блок виділення ФК включає два рівні арбітражу. Для кожного вхідного ФК відповідний арбітр першого рівня здійснює попередній вибір одного з вільних ФК у транку призначення. Лінія статусу вихідного ФК переходить в нуль (ФК зайнятий), коли канал виділяється одному з вхідних ФК і встановлюється в одиницю (ФК вільний), коли останній фліт пакету покидає маршрутизатор через зайнятий ФК. Інформація про статус каналів транку призначення надходить до відповідного арбітра першого рівня через виділений мультиплексор (одному арбітру відповідає один мультиплексор). Виходи арбітрів першого рівня підключені до входів арбітрів другого рівня за допомогою демультиплексорів (рис.3), причому *k*-й вихід арбітра першого рівня відповідає *p*-му входу *q*-го арбітра другого рівня. Значення p і q можна вирахувати за допомогою наступних рівнянь:

$$p = M \cdot i + j\,, \quad q = M \cdot t + k\,,$$

де числа i та j позначають номера вхідного транка та вхідного ФК, що відповідають арбітру першого рівня; M дорівнює кількості ФК у транках маршрутизатора; числа t і k індексують транк призначення та номер ФК на виході маршрутизатора (який раніше був вирахуваний арбітром першого рівня). Можлива ситуація, коли два або більше арбітрів першого рівня оберуть один і той же вихідний ФК. Для остаточного вибору вхідного ФК, який буде з’єднаний з певним вихідним каналом, використовується другий рівень арбітражу, де кожному вихідному ФК відповідає арбітр з кількістю входів, що дорівнює числу вхідних ФК. Активний вихід арбітра другого рівня вказує на номер вхідного ФК, якому виділений даний вихідний ФК. Для реалізації арбітрів обрано матричний метод (matrix arbiter), що забезпечує більшу пропускну здатність, ніж арбітр з циклічним зсувом пріоритетів (round-robin arbiter) [21].

Просування фліта

Для просування фліта через встановлене з’єднання необхідна наявність даних у відповідній черзі вхідного ФК і ненульове число кредитів у ФК на виході маршрутизатора. В разі виконання зазначених умов відповідний фліт вилучається з черги на вході і через встановлене з’єднання передається наступному маршрутизатору. При цьому надсилається

кредит попередньому вузлу мережі, інформуючи його про те, що у вхідній черзі звільнилася одиниця місця. У той же час число кредитів для ФК на виході, через який щойно було передано фліт, декрементується.

Результати симуляції і синтезу RTL моделі маршрутизатора для МнК з АК

Для оцінки транспортної затримки та пропускної здатності wormhole МнК з АК на основі запропонованого вище маршрутизатора авторами даної праці створено RTL модель такої МнК і проведено моделювання її роботи (абревіатура RTL – це скорочення від “register transfer level” і застосовується для характеристики моделі ІМС, з якої за допомогою спеціалізованих САПР можна синтезувати апаратурну реалізацію мікросхеми). В якості мови опису пропонованої RTL моделі обрано System Verilog. Аналіз результатів симуляції роботи такої моделі наведено в [11, 12] і свідчить, що використання АК в wormhole МнК дозволяє підвищити поріг насичення класичної wormhole мережі (без ВК) на 300%. У порівнянні з МнК на основі ВК запропонований підхід призводить до зростання порогу насичення на 126% (МнК Netmaker [22]) і на 152% (МнК Hermes[14]).

З метою оцінки апаратурних витрат синтезовано кілька конфігурацій маршрутизаторів для МнК з АК в САПР Quartus II. В якості цільової мікросхеми обрана FPGA EP4SGX230KF40C2 сімейства Stratix IV. Результати викладені в [12]. Загалом, wormhole маршрутизатор для МнК з АК вимагає для своєї реалізації такої ж кількості апаратурних ресурсів, як і його аналог з ВК. Наприклад, ВК маршрутизатор з бібліотеки Netmaker, що містить п’ять двонаправлених портів шириною 32 біти кожен, довжиною черг 4 фліти і двома ВК на порт займає 2400 таблиць істинності, 2232 тригери і має максимальну частоту роботи 130 МГц. З іншого боку, запропонований маршрутизатор з АК в такій же самій конфігурації, але з двома ФК на транк, шириною 16 бітів кожен, займає 2167 таблиць істинності, 1074 тригери і може функціонувати на частоті 190 МГц.

3. Обгрунтування вибору мови опису високопродуктивної моделі wormhole маршрутизатора з АК

З метою зменшення кількості апаратурних ресурсів, необхідних для реалізації МнК з АК, в [12] пропонується проводити параметричну оптимізацію кількості ФК всередині агрегованих логічних з’єднань мережі. Виконання оптимізації такого роду потребує швидкодіючої моделі об’єкту дослідження. В поточному і наступних розділах буде описано процес створення подібної моделі.

В [23] показано, що незважаючи на мову опису апаратури (SPICE, Verilog, VHDL, SystemC), яка використовується для створення моделі ІМС, швидкість виконання симуляції визначається ступенем детальності опису моделі. При здійсненні дискретно-подієвого моделювання симулятор виконує наступні дії: передачу даних, модифікацію черги запланованих подій, виконання операцій над даними. Зі збільшенням числа модулів, міжмодульних зв’язків та обсягів даних, що передаються через них, зростає обсяг необхідних для виконання операцій, а також ускладнюється робота з чергою подій, що закономірно приводить до збільшення тривалості моделювання. Таким чином, для прискорення симуляції необхідно зменшувати рівень деталізації опису, кількість модулів і число міжмодульних з’єднань. Щоб уникнути зменшення точності моделювання слід опускати лише ті деталі, які не впливають або слабо впливають на результат симуляції. Це досягається використанням методу абстрагування.

Зменшення тривалості симуляції моделі ІМС шляхом збільшення рівня її абстракції можливе при використанні будь-якої з мов опису апаратури (HDL – hardware description language), проте мова SystemC має наступні переваги, що обумовлюють її вибір з цією метою:

– systemC має безкоштовний і відкритий вихідний код;

– systemC реалізований як бібліотека класів на C++, що дозволяє використовувати весь розвинений інструментарій об’єктно-орієнтованого програмування цієї мови для підвищення рівня абстракції моделі;

– з попереднього пункту випливає можливість використання в моделях на SystemC величезного масиву існуючих ефективних і відпрацьованих рішень, створених на мовах C та C++ (STL, Boost [24]);

– можливість синтезу апаратурних рішень як по низькорівневому, так і по високорівневому опису моделі (Catapult [25]);

– можливість ко-симуляції з кодом, створеним за допомогою інших HDL (Verilog, System Verilog, VHDL);

– сучасні симулятори, такі як ModelSim, компілюють модель, описану за допомогою одного з HDL, в машинно-незалежний об’єктний код, який потім оптимізується і далі виконується сам процес моделювання. При зміні одного з параметрів моделі необхідно виконувати заново процедури компіляції і оптимізації, що займає від одиниць до десятків хвилин. Опис моделі ІМС за допомогою SystemC дозволяє динамічно створювати в пам'яті параметризовані модулі системи без перекомпіляції моделі;

– при виконанні параметричної оптимізації моделі на SystemC симуляція та оптимізація виконуються однією програмою, написаною на C++. Немає необхідності у передачі великих обсягів проміжної інформації між програмою оптимізатором і програмою симулятором, як у випадку використання, наприклад, ModelSim.

4. Реалізація високопродуктивної моделі маршрутизатора для МнК з АК

Для проведення параметричної оптимізації кількості ФК всередині агрегованих логічних каналів МнК необхідна інформація про затримки передачі пакетів і коефіцієнти використання транків. Немає необхідності в інформації про таймінги, тому можна використовувати поведінкову модель, що реалізує алгоритм функціонування пристрою без урахування затримок розповсюдження сигналів всередині ІМС.

Загальний підхід

Під час створення поведінкової, швидкодіючої моделі інтегрального маршрутизатора з АК на SystemC авторами використані наступні положення:

– застосування вбудованих типів даних C++ (*int*, *bool*) замість похідних типів, притаманних SystemC (*sc_bit, sc_logic, sc_bv*), призводить до зменшення тривалості моделювання. Наприклад, для опису сигналів двухсимвольний тип *bool* обчислювально ефективніший, ніж чотирьох символьний тип *sc_logic* ('0 ', '1', 'x', 'z'). Багатобітові сигнали бажано описувати за допомогою типу *int* та структур даних замість масивів однобітових змінних [23];

– в SystemC процеси типу SC_METHOD моделюються швидше процесів SC_THREAD, оскільки останні мають власний стек та локальні змінні, що вимагає виконання додаткових операцій для обслуговування контексту потоку [23];

– в SystemC для передачі інформації про зміну сигналу від одного процесу до іншого через відповідні порти та з’єднання потрібно викликати три функції і виконати три операції копіювання. Мінімізація міжмодульних з’єднань призводить до зменшення інтенсивності викликів згаданих функцій і, як наслідок, до зменшення тривалості моделювання [23];

– для зменшення числа модулів і міжмодульних зв’язків кожен маршрутизатор в МнК з АК описаний за допомогою процесу типу SC_METHOD, що має інтерфейси (порти) для підключення сигналів. Всі підмодулі всередині маршрутизатора та зв’язки (сигнали) між ними замінені послідовною програмою, що реалізує алгоритм функціонування маршрутизатора, використовуючи вбудовані типи даних C++ і високорівневі контейнери (наприклад, буфери маршрутизатора реалізовані за допомогою *deque* з STL).

Типи даних та змінні

Представимо алгоритм програми, що моделює процес функціонування інтегрального маршрутизатора з АК. Почнемо з огляду використовуваних змінних. Перш за все – це вхідні та вихідні порти маршрутизатора: *clk, rst, cntrl_in, cntrl_out, flits_in, flits_out*. Входи *clk* і *rst* мають тип *sc_in<bool>*. Процедура, що моделює роботу маршрутизатора, викликається по передньому фронту сигналу синхронізації *clk*. Вхід *rst* є сигналом скидання і початкової

ініціалізації моделі. Вхідний інтерфейс *cntrl_in* являє собою масив змінних типу *sc_in<bool>*, розмірністю NxM, де M – число ФК в кожному з N агрегованих логічних з'єднань, підключених до маршрутизатора. Кожен елемент масиву *cntrl_in* являє собою вхід управління потоком, на який надходять кредити від маршрутизатора-приймача даних. Вище написане справедливе і для *cntrl_out* з урахуванням того, що елементи цього масиву є виходами і мають тип *sc_out<bool>*. Вхідний інтерфейс, через який в маршрутизатор надходять фліти, описується масивом *flits_in*, розмірність якого NxM, а елементи мають тип *sc_in<flit_t>*. Структура *flit_t* містить поля *valid* (*bool*), *head* (*bool*), *tail* (*bool*), *output_trunk* (*int*), поле переданих даних *data* (int), а також поля для додаткової маршрутної інформації, від якої можна абстрагуватися при розгляді даного алгоритму. Розмірність масиву *flits_out* – NxM, а тип елементів – *sc_out<flit_t>*. Решта змінних, які використовуються в програмі, розглядаються по ходу опису алгоритму і являють собою масиви розмірністю NxM. Практично всі згадані масиви мають тип елементів *bool*, окрім *allocated_pl* (*int*), *credit_count* (*int*) і *output_trunk* (*int*).

Алгоритм

Розглянемо кроки, з яких складається алгоритм функціонування моделі інтегрального маршрутизатора з АК:

1. Якщо сигнал скидання *rst* дорівнює одиниці, то виконати процедуру, що відповідає за початкову ініціалізацію змінних, інакше перейти до наступного кроку. Процедура ініціалізації обнуляє всі змінні типу *int*, що використовуються в програмі, а змінним типу *bool* присвоює значення *false*. Вище написане справедливе для всіх змінних, окрім масивів *pl_status* та *credit_count*. Всі елементи *pl_status* повинні бути встановлені в значення *true*, а всі елементи *credit_count* повинні дорівнювати розміру черги в системі;

2. $\forall i, j : i \in \overline{[1; N]}, j \in [1; M]$, де j індексує ФК в i-му вхідному транку, якщо *input_trunk_pop[i][j] == true* і біт *tail* фліта, що знаходиться в голові черги *fifo[i][j]* приймає значення *true* (останній фліт пакету), виконати: *allocated_pl_valid[i][j] = false*;

3. $\forall i, j : i \in [1; N], j \in [1; M]$, де j індексує ФК в i-му вхідному транку, якщо *input_trunk_pop[i][j] == true*, то вилучити фліт з голови черги *fifo[i][j]*;

4. $\forall k, p : k \in [1; N], p \in [1; M]$, де p індексує ФК в k-му вихідному транку, якщо *flits_out_tail[k][p] == true* і *flits_out_valid[k][p] == true*, позначити відповідний вихідний канал як вільний, виконавши *pl_status[k][p] = true*;

5. $\forall i, j : i \in [1; N], j \in [1; M]$, де j позначає номер ФК в i-му вхідному транку, якщо черга *fifo[i][j]* не порожня і *allocated_pl_valid[i][j] == true,* і *credit_count[k][p] != 0*, виконати *input_trunk_pop[i][j] = true*, інакше – *input_trunk_pop[i][j] = false*, де *k = out_trunk[i][j]*, *p = allocated_pl[i][j]*. На цьому кроці визначається, з яких вхідних черг будуть вилучені фліти на наступному циклі тактової частоти;

6. $\forall i, j : i \in [1; N], j \in [1; M]$, де i та j індексують сигнал вихідного кредиту для j-го ФК i-го вхідного транку, виконати *cntrl_out[i][j] = input_trunk_pop[i][j]*;

7. $\forall i, j : i \in [1; N], j \in [1; M]$, де j індексує ФК в i-му вхідному транку, якщо *allocated_pl_valid[i][j] == true*, то виконати наступні підпункти:

7.1. Скопіювати фліт, що знаходиться в голові черги *fifo[i][j]*, в змінну *tmp_flit*;

7.2. Якщо в змінній *tmp_flit* знаходиться перший фліт пакету (*tmp_flit.head == true*), виконати look-ahead маршрутизацію, визначивши транк призначення для наступного вузла мережі за допомогою виклику функції *route()*: *tmp_flit = route(tmp_flit)*. Дана операція оновлює маршрутну інформацію змінної *tmp_flit*. Деталі реалізації look-ahead маршрутизації описані в [21];

7.3. Модифікувати біт *valid* змінної *tmp_flit*: *tmp_flit.valid = input_trunk_pop[i][j];*

7.4. Записати в вихідний порт *flits_out[k][p]* змінну *tmp_flit*: *flits_out[k][p] = tmp_flit*, де *k = out_trunk[i][j]*, *p = allocated_pl[i][j]*;

7.5. Виконати *flits_out_valid[k][p] = tmp_flit.valid*, де *k = out_trunk[i][j]*, *p = allocated_pl[i][j]*;

7.6. Виконати *flits_out_tail[k][p] = tmp_flit.tail*, де *k = out_trunk[i][j]*, *p = allocated_pl[i][j]*;

7.7. Якщо *tmp_flit.valid == true,* виконати *credit_count[k][p] = credit_count[k][p] – 1*, де *k = out_trunk[i][j]*, *p = allocated_pl[i][j]*;

8. $\forall i, j : i \in [1; N], j \in [1; M]$, де *j* індексує ФК в *i*-му вхідному транку, якщо *flits_in[i][j].valid == true*, то помістити фліт з вхідного порту *flits_in[i][j]* в хвіст черги *fifo[i][j]*;

9. $\forall i, j : i \in [1; N], j \in [1; M]$, де *j* індексує ФК в *i*-му вхідному транку, якщо в голові черги *fifo[i][j]* знаходиться перший фліт пакету (*fifo[i][j].front().head == true*), зберегти маршрутну інформацію для даного пакету: *out_trunk[i][j] = fifo[i][j].front().output_trunk;*

10. $\forall i, j : i \in [1; N], j \in [1; M]$, де *j* індексує ФК в *i*-му вхідному транку, сформувати запити на виділення ФК у вихідних транках маршрутизатора, виконавши наступне: якщо черга *fifo[i][j]* не порожня і *allocated_pl_valid[i][j] == false*, то присвоїти елементу масиву *alloc_req[i][j]* значення *true* , інакше – *false*;

11. Викликати функцію *allocate(...)*, що реалізує алгоритм функціонування блоку виділення ФК, структурна схема якого показана на рис.3. Задача функції *allocate(...)* полягає у виділенні ФК у вихідних транках маршрутизатора вхідним ФК, які подали заявку на попередньому кроці алгоритму. Вхідні дані функції: масиви *alloc_req*, *out_trunk* і *pl_status*, вихідні дані: масиви *allocated_pl*, *allocated_pl_valid* і *allocated_pl_out*. Для кожного вхідного ФК елемент масиву *alloc_req* визначає заявку на виділення ФК у вихідному транку, елемент масиву *out_trunk* визначає номер транка призначення для даного вхідного ФК, елемент масиву *allocated_pl* визначає номер виділеного ФК, а значення елемента масиву *allocated_pl_valid*, рівне *true*, сигналізує про успішне виділення ФК для даного входу. Для кожного вихідного ФК елемент масиву *pl_status* вказує на статус ФК (*true* – вільний, *false* – зайнятий, тобто уже виділений якому-небудь з вхідних ФК протягом попередніх кроків алгоритму). Значення *true* елемента масиву *allocated_pl_out* показує, що даний вихідний ФК виділений одному з вхідних ФК в результаті виконання функції *allocate(...)* на поточному кроці;

12. $\forall k, p : k \in [1; N], p \in [1; M]$, де *p* індексує ФК в *k*-му вихідному транку, якщо *allocated_pl_out[k][p] == true*, позначити відповідний вихідний канал як зайнятий, виконавши *pl_status[k][p] = false*;

13. $\forall k, p : k \in [1; N], p \in [1; M]$, де *k* і *p* індексують вхідний сигнал кредиту, що відповідає *p*-му ФК в k-му вихідному транку, якщо *cntrl_in[k][p] == true*, виконати *credit_count[k][p] = credit_count[k][p] + 1* .

5. Аналіз результатів моделювання

Далі буде проведено аналіз продуктивності і точності високорівневої поведінкової моделі МнК з АК на SystemC у порівнянні з її RTL аналогом на System Verilog. На основі описаної вище поведінкової моделі маршрутизатора з АК на SystemC було створено МнК з матричною (mesh) топологією розмірністю 8x8. Кожен маршрутизатор в такій мережі за допомогою чотирьох двонаправлених транків з’єднаний з сусідніми вузлами у відповідності до обраної топології. П’ята пара транків використовується для зв’язку маршрутизатора з генератором-приймачем трафіку, що імітує функціонал обчислювального модуля. Довжина буферних черг для кожного вхідного ФК складає чотири фліта. Компіляція бібліотеки SystemC і розробленої на її основі моделі МнК з АК проводилась з використанням компілятора g++, що входить до складу GCC версії 4.3.4. Запуск моделі виконувався в середовищі Cygwin 1.7.7 на ПК, що включає процесор Intel Core 2 Duo T5300 і 2Гб ОЗУ, під управлінням ОС Windows XP SP3.

Для визначення проміжних вузлів на шляху просування пакетів обрано метод покординатної статичної XY маршрутизації, яка широко застосовується в МнК. Під час покординатної маршрутизації передача здійснюється спочатку вздовж горизонтального напрямку двомірної решітки (mesh), а потім вздовж її вертикального напрямку [21]. Протягом одного прогону моделі генератори трафіку створюють пакети, що складаються з флітів і з інтенсивністю λ вводять ці фліти в мережу. Значення λ лежить на проміжку від нуля до одиниці і показує, скільки в середньому флітів вводиться в мережу певним обчислювальним модулем (вузлом) за цикл тактової частоти. Інтервали часу між інжекцією флітів в МнК розподілені за експоненціальним законом і в середньому дорівнюють 1/λ. Для збереження характеристик часового розподілу створені пакети перед введенням в мережу буферизуються у черзі [21]. При цьому генератори трафіку інжектують фліти в МнК лише через один ФК. Зчитування ж флітів приймачами трафіку відбувається через всі ФК відповідного транку. Для опису просторового розподілу пакетів обрано рівномірно-випадковий закон, коли для будь-якого джерела ймовірність вибору того чи іншого приймача однакова. При цьому для кожного наступного пакета генерується новий пункт призначення. Після початку симуляції всі вузли в мережі генерують по 1100 пакетів, а збір результатів моделювання починається після прийому перших ста повідомлень. Це робиться для забезпечення переходу МнК в усталений режим [21]. Моделювання закінчується після реєстрації в пунктах призначення всіх відправлених повідомлень.

На рис.4 показано залежність транспортної затримки (в циклах тактової частоти) від прикладеного навантаження (інтенсивності інжекції флітів в мережу λ) для різних конфігурацій високорівневої моделі МнК з АК на SystemC та її деталізованого RTL аналога на System Verilog. Для кожної з моделей досліджено по три конфігурації МнК – з одним, двома і чотирма ФК на транк. Результати симуляції RTL моделі МнК з АК на System Verilog отримані авторами раніше в роботі [12].

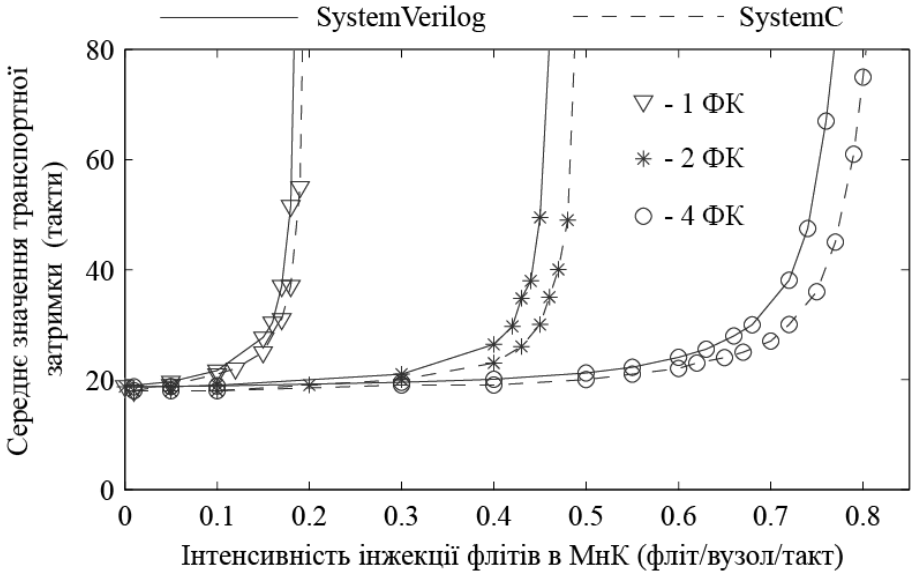

Рис.4. Залежність транспортної затримки від інтенсивності інжекції флітів для RTL моделі МнК з АК на System Verilog і її високопродуктивного аналога на SystemC: ФК – кількість фізичних каналів всередині транків МнК з АК

З рис.4 видно, що до моменту різкого зростання кривих залежності транспортної затримки від прикладеного навантаження результати функціонування обох моделей практично ідентичні (помилка <1%). В момент, коли настає насичення мережі, помилка швидкодіючої SystemC моделі різко зростає, що пов’язано з експоненціальним характером росту залежностей на рис.4. Діапазон інтенсивності прикладеного навантаження, на якому помилка оцінки транспортної затримки збільшується на 50% і вище, визначає точність передбачення порогу насичення мережі і становить 5.6% від порогу насичення для конфігурації МнК з 1 ФК/транк, 6.1% для конфігурації МнК з 2 ФК/транк і 3.9% для

конфігурації МнК з 4 ФК/транк. Автори даної роботи вважають, що зменшення точності симуляції високорівневої SystemC моделі обумовлено підвищенням рівня її абстракції. Незважаючи на це, пропонована модель може бути використана на початкових етапах параметричної оптимізації МнК з наступною верифікацією отриманих результатів на більш точній моделі вентильного рівня абстракції.

Середній час моделювання та об’єм використаної пам’яті для різних конфігурацій досліджуваних моделей наведено в табл. 1. При заданому числі пакетів, які необхідно ввести в мережу за один прогін моделі, тривалість симуляції залежить від величини прикладеного до мережі навантаження, зменшуючись у міру збільшення останнього. Значення тривалості симуляції з табл.1 отримані усередненням результатів прогонів моделі для різних значень прикладеного навантаження (від нуля до порогу насичення) при одній і тій же конфігурації мережі. Вимірювання тривалості одного прогону моделі виконувалось за допомогою утиліти time, що входить до складу пакету Cygwin. Визначення об’єму пам’яті, яку займає модель, здійснювалося за допомогою диспетчера задач Windows.

Таблиця 1

Тривалість симуляції (час) і обсяг використаної пам’яті (пам.) для моделей МнК з АК на System Verilog та SystemC

Тип моделі	Характеристики моделі					
	1 ФК/транк		2 ФК/транк		4 ФК/транк	
	Час, сек.	*Пам., Кбайт*	*Час, сек.*	*Пам., Кбайт*	*Час, сек.*	*Пам., Кбайт*
RTL на System Verilog	1 061	139 928	1 102	278 740	1 710	714 728
Високорівнева на SystemC	268	4 760	146	5 156	167	5 908
Виграш у продуктивності	3.9	29.4	7.5	54	10.2	121

З табл.1 видно, що запропонована високорівнева модель на SystemC забезпечує значне зменшення тривалості моделювання (від 3.9 до 10.2 разів) та обсягу займаної пам’яті (від 29.4 до 121 разів) в порівнянні зі своїм деталізованим RTL аналогом на System Verilog.

Висновки

В роботі досліджено запропоновану авторами високопродуктивну модель маршрутизатора для МнК з АК, описану з використанням SystemC. З результатів симуляції випливає, що для розробленої моделі помилка передбачення порогу насичення не перевищує 6.1%, у порівнянні з її RTL аналогом на System Veilog. При цьому тривалість симуляції скорочується на порядок, а обсяг займаної пам’яті зменшується в 121 разів. Такі результати досягнуто зменшенням кількості модулів апаратурного рівня абстракції та зв’язків між ними, застосуванням вбудованих типів даних мови C++, а також використанням SystemC процесів типу SC_METHOD замість SC_THREAD.

Вектор подальших досліджень спрямований на розробку методу параметричної оптимізації кількості ФК в агрегованих логічних з’єднаннях МнК с АК на базі розробленої високопродуктивної моделі на SystemC. Пріоритетним також є пошук шляхів зниження помилки моделі на SystemC у порівнянні з її RTL аналогом на System Verilog.

ЛІТЕРАТУРА

1. Bjerregaard T., Mahadevan S. A survey of research and practices of network-on-chip // ACM Computing Surveys.– 2006.– Vol.38, №1.– P.1 – 51.

2. Angiolini F., Meloni P., Benini L. A layout-aware analysis of networks-on-chip and traditional interconnects for mpsocs // IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems. – 2007. – Vol.26, №3. – P. 421 – 434.

3. Lee H.G., Ogras U.Y., Marculescu R. On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus and network-on-chip approaches // ACM Transactions on Design Automation of Electronic Systems.– 2007.– Vol.12, №3. – P. 1 – 20.

4. Dally W., Towles B**.** Route packets, not wires: on-chip interconnection networks // Proceedings of the 38th annual Design Automation Conference (June 2001). – Las Vegas, USA. – P.684-689.

5. Atienza D., Angiolini F., Benini L. Network-On-Chip Design and Synthesis Outlook // Integration The VLSI journal.– 2008.– Vol.41, №3.– P.340 – 359.

6. Marculescu R., Bogdan P. The Chip Is the Network: Toward a Science of Network-on-Chip Design // Foundations and Trends in Electronic Design Automation. – 2009. – Vol.2, №4. – P.371 – 461.

7. Marculescu R., Ogras U. Outstanding Research Problems in NoC Design: System, Microarchitecture and Circuit Perspectives // IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems.– 2009.– Vol.28, №1.– P.3 – 21.

8. Gu H. Survey of dynamically reconfigurable Network-on-chip // in Proc. of International Conference on Future Computer Sciences and Application (June 2011).– Hong Kong, China. – P. 200 – 203.

9. Dally W.J. Performance analysis of k-ary n-cube interconnection networks // IEEE Transactions on Computers.– 1990.– Vol.39, №6.– P.775 – 785.

10. Dally W.J. Virtual-channel flow control // IEEE Transactions on Parallel and Distributed Systems.– 1992.– Vol.3, №2.– P.194 – 205.

11. Korotkiy E.V., Lysenko O.M., Tereshin M.O. Link aggregation in networks-on-chip // Прикладная радиоэлетроника.– 2011.– том 10, № 3.– С. 330 – 336.

12. Korotkyi I., Lysenko O. Hardware implementation of link aggregation in networks-on-chip // in Proc. of World Congress on Information and Communication Technologies (Dec.2011).– Mumbai, India.– P.1112 – 1117.

13. Dally W.J., Peh L.S. A Delay Model and Speculative Architecture for Pipelined Routers // in Proc. of 7-th International Symposium of High-Performance Comp. Arch. (20-24 Jan. 2001).– Nuevo Leone, Mexico.– P.255 – 266.

14. Mello A., Calazans N., Moraes F. Virtual channels in networks on chip: implementation and evaluation on Hermes NoC // in Proc. of 18th Symp. Integr. Circ. and Syst. Design (2005).– New York, USA.– P.178 – 183.

15. Janarthanan A. Networks-on-chip based high performance communication architectures for FPGAs: Ph.D. dissertation.– Dept. Elect. Comp. Eng. and Comp. Science, Univ. of Cincinnati, Cincinnati, Ohio, 2008.– 127p.

16. Mullins R., West A., Moore S. Low-latency virtual-channel routers for on-chip networks // in Proc. of 31-th Intern. Symp. on Comp. Arch (June 2004).– Munich, Germany.– P.188 – 197.

17. Короткий Е.В., Лысенко А.Н. Влияние виртуальных каналов на транспортную задержку сети на кристалле // Проблеми інформатизації та управління.– 2011.– №4. – С. 69 – 73.

18. Альтшуллер Г.С. Алгоритм изобретения.– М.: Моск. рабочий, 1973.– 296с.

19. Link aggregation: IEEE Standart 802.3ad, 2000.– 173p.

20. IEEE Standart for Local and Metropolitan Area Networks – Link Aggregation: IEEE Standart 802.1ax, 2008.– 145p.

21. Dally W.J., Towles B. Principles and Practices of Interconnection Networks.– San Francisco: Morgan Kaufmann Publishers, 2004.– 550p.

22. Netmaker.– Режим доступу: http://www-dyn.cl.cam.ac.uk /~rdm34/wiki

23. Black D., Donovan J. SystemC: from the ground up.– Boston: Kluwer Academic Publishers, 2004.– 244p.

24. Boost C++ libraries.– Режим доступу: http://www.boost.org/

25. Catapult C synthesis.– Режим доступу: http://www.mentor.com/esl/catapult.