ПРОБЛЕМИ ПРИЙНЯТТЯ РІШЕНЬ І УПРАВЛІННЯ В ЕКОНОМІЧНИХ, ТЕХНІЧНИХ, ЕКОЛОГІЧНИХ І СОЦІАЛЬНИХ СИСТЕМАХ

УДК 004.383
DOI: 10.20535/SRIT.2308-8893.2016.1.06

# СКЛАДАННЯ РОЗКЛАДУ ДЛЯ ГРАФІВ СИНХРОННИХ ПОТОКІВ ДАНИХ

**А.М. СЕРГІЄНКО, В.П. СІМОНЕНКО**

Розглянуто задачу складання розкладу для алгоритму, який заданий графом синхронних потоків даних (ГСПД). Запропоновано метод складання періодичного розкладу ГСПД з періодом $L$ тактів, оснований на перетворенні його у просторовий ГСПД, вершини якого мають координати місця та моменту виконання відповідних операторів алгоритму. На координати просторового ГСПД накладено обмеження: оператори, які виконуються в одному процесорному елементі, не повинні мати однакові такти свого виконання, які взято за модулем $L$. Завдяки цьому ГСПД відображається у спеціалізований обчислювач, який виконує алгоритм у конвеєрному режимі з оптимізованою завантаженністю ресурсів. Показано алгоритм пошуку субоптимального розкладу на основі просторового ГСПД.

## ВСТУП

У персональних комп'ютерах, засобах мобільного зв'язку, пристроях цифрового оброблення сигналів та багатьох інших реалізовано програми і спецпроцесори, які виконують обчислювальні процеси, що повторюються з періодом, який збігається з інтервалом надходження вхідних даних. Ці процеси виконуються за алгоритмами, які обробляють дані, організовані у потоки. Такі алгоритми доцільно задавати на моделі графу потоків даних. Граф потоку даних — це напрямлений граф, вершини якого — актори — виражають операції, а дуги — потоки, через які передаються дані. У праці [1] запропоновано класифікацію різних моделей графу потоку даних, серед яких виділяються графи синхронних потоків даних (ГСПД, Synchronous Dataflow — SDF). У ГСПД кожним актором під час виконання алгоритму генерується і використовується кількість змінних, які є незмінними від циклу до циклу [2, 3].

Розрізняють однорідні ГСПД (homoheneous SDF) і неоднорідні ГСПД (multirate SDF). В останніх кількість змінних, які використані та згенеровані кожною вершиною протягом одного циклу, може бути більшою за одиницю, унаслідок чого вони мають більш компактну форму задання алгоритму [4]. Для спрощення аналізу неоднорідного ГСПД та на його основі синтезу обчислювальних пристроїв граф найчастіше перетворюють в еквівалентний однорідний ГСПД [4, 5]. У роботі розглядаються лише однорідні ГСПД.

Граф синхронних потоків даних має пряму аналогію з певним обчислювальним пристроєм. Логічні блоки такого пристрою відповідають вершинам

ГСПД, лінії зв'язку — дугам графу, а регістри операндів — затримкам у дугах. Тому ГСПД часто використовують для одиничного відображення алгоритму у структуру спеціалізованого обчислювального пристрою, яка часто буває неоптимальною. Для пошуку оптимального структурного або програмного розв'язку необхідно скласти розклад виконання алгоритму в моделі ГСПД. У роботі розглядається метод складання розкладу на основі просторового ГСПД, який забезпечує швидкий пошук як ефективного розкладу, так і структури обчислювального пристрою.

## СКЛАДАННЯ РОЗКЛАДУ ДЛЯ ГСПД

Розглянемо приклад тестового алгоритму обчислення диференціального рівняння $y'' + bxy' + cy = 0$ [6,7]. Його розв'язують за алгоритмом:

```
i = 0;
while (x_i < a) do
   x_{i+1} = x_i + dx;
   u_{i+1} = u_i – b*x_i*u_i*dx – c*y_i*dx;                    (1)
   y_{i+1} = y_i + u_i*dx;
   i = i + 1;
end;
```

де $i$ — номер ітерації; $a$, $b$, $c$, $dx$ — константи. Алгоритм зупиняється, тобто переривається запис у регістри змінних $x_i$, $u_i$, $y_i$ із досягненням межі $x_i \geq a$.

Цей алгоритм зображено як ГСПД на рис. 1. На ньому знаками «+», «×», кружечком і товстою крапкою позначено вершини множення на коефіцієнт, додавання, подання $(i+1)$-х та $i$-х змінних відповідно. Пунктирними стрілками позначено дуги міжітераційної залежності, які навантажені затримками, позначеними товстими відрізками.

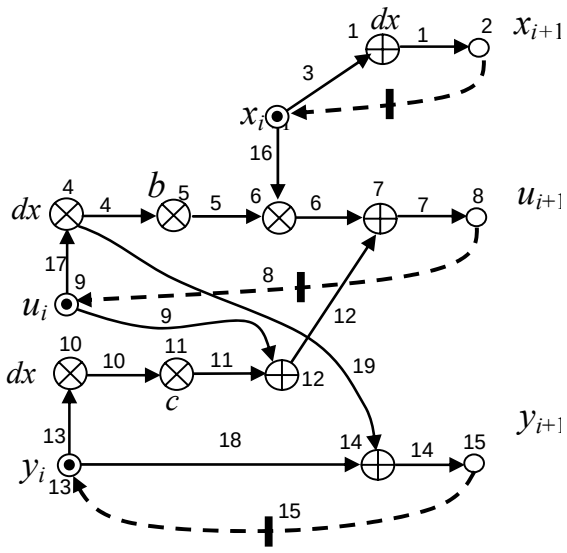


*Рис. 1.* Граф синхронних потоків розв'язання диференціального рівняння

Змінні у ГСПД позначаються мітками. Виконання алгоритму в ГСПД полягає у переміщенні міток уздовж дуг, спрацюванні вершин, якщо на їх входах є мітки, та видачі цих міток на виходи вершин. Мітка в дузі, яка навантажена затримками, затримується в ній на відповідну кількість ітерацій. Чергова ітерація закінчується в момент, коли мітки повертаються у початкові вершини, наприклад, позначені на рис. 1 порожнім кружечком.

За одиничного відображення ГСПД у структуру пристрою цикл виконання однієї ітерації дорівнює од-

ному такту. При цьому розклад виконання алгоритму визначається поведінкою ГСПД та затримками логічних схем, у які відображаються вершини графу. Але через велику довжину критичного шляху і великі апаратні витрати такий розклад не є раціональним. У цьому прикладі довжина критичного шляху дорівнює $T_C = 3t_M + t_A$, де $t_M$, $t_A$ — затримки помножувача та суматора відповідно.

Пошук можливих розкладів виконання ГСПД є одним з етапів задачі синтезу конвеєрного обчислювального пристрою, який найчастіше розв'язують шляхом виконання, крім етапу розкладу, етапів вибору множини ресурсів, призначення операцій на ресурси, побудови структури обчислювача і його блока керування, а також вибору обчислювального пристрою, який є оптимальним за заданим критерієм.

Найбільш відомі та вживані методи складання розкладу, які мають обмеження на ресурси або час виконання з урахуванням ациклічного алгоритму, що виконується одноразово. У розгляданому випадку — це ациклічний підграф ГСПД. Це такі методи, як метод спискового планування [9], складання розкладу напрямленим зусиллям [10]. Для конвеєрних обчислювальних пристроїв у працях [11, 12] запропоновано методи, у яких після спискового складання розкладу призначаються оператори на конвеєрні ресурси з урахуванням циклічного характеру обчислень. Використовують також метод відображення гнізда циклів у неконвеєрний обчислювальний пристрій з подальшим перетворенням його у конвеєрний пристрій [13]. Розклад завантаження регістрів обчислювального пристрою ефективно складати згідно з евристикою Тсенга [14] та методом лівої межі (left-edge sheduling) [15].

Для урахування циклічності виконання алгоритму розклад часто складають з використанням інтервального графу з циклічними дугами [16]. Для спрощення розв'язання задачі планування використовують різні евристики [17, 18] та методи ресинхронізації (retiming) [18, 19]. У праці [20] запропоновано побудувати періодичний граф залежності за даними, розмістити його у багатовимірному просторі як напрямлений граф і знаходити розклад як проекцію графу на часову вісь.

Складність багатоетапного синтезу обчислювального пристрою полягає в тому, що різні аспекти синтезу і етапи розроблення обчислювального пристрою — етапи вибору множини ресурсів, складання розкладу і призначення операцій на ресурси — істотно залежать один від одного. Кожен з етапів проектування не може бути виконаний незалежно так, щоб не зменшити можливості глобальної оптимізації обчислювального пристрою. Наприклад, мінімізація апаратних витрат при виборі ресурсів суперечить мінімізації тривалості циклу алгоритму для складання розкладу. Крім того, якщо за одиничного відображення алгоритму в структуру складність синтезу обчислювального пристрою оцінюється поліномом, то при відображенні з уповільненням у $L$ разів задача синтезу стає NP-повною [8].

## ПРОСТОРОВИЙ ГСПД ТА ЙОГО РОЗКЛАД

Задання розкладу для ГСПД означає призначення його вершинам моментів спрацювання. Структуру обчислювального пристрою можна отримати шляхом гомоморфного перетворення ГСПД у граф структури. Таке перетворення

означає склеювання вершин, оператори яких виконуються в одному процесорному елементі (ПЕ) пристрою, але в різні моменти часу [20].

Гомоморфне перетворення графу виконують, призначивши вершинам теги з номерами вершин ПЕ, де вони будуть виконуватися. Тоді у вершину ПЕ склеюються вершини з однаковими номерами. Отже, вершина ГСПД повинна мати тег з номером ПЕ, а також з моментом виконання оператора та його типом.

Теги вершин зображаються цілочисловими багатовимірними векторами $\mathbf{K}_i$. При цьому момент спрацювання задається як номер такту. Тоді ГСПД буде поданий в багатовимірному цілочисловому просторі. Відображення алгоритму полягає у призначенні елементам матриці $K$ цих векторів певних значень і побудові структури пристрою з обчисленням критерію його оптимальності. Тоді оптимізація обчислювального пристрою полягає в побудові множини матриць $K$ і виборі найбільш пріоритетної матриці за критерієм оптимальності.

У праці [21] описано метод синтезу обчислювального пристрою, у якому ГСПД подано у тривимірному цілочисловому просторі у вигляді конфігурації алгоритму $K_G = (K, D, A)$, де $K$ — матриця векторів-вершин $\mathbf{K}_i$, що відповідають операторам алгоритму; $D$ — матриця векторів-дуг $\mathbf{D}_j$, які виражають безпосередні інформаційні зв'язки між операторами; $A$ — матриця інцидентності ГСПД. У векторі-вершині $\mathbf{K}_i = (k_i, s_i, t_i)^{\mathrm{T}}$ координати $k_i, s_i, t_i$ відповідають типу оператора, номеру процесорного елемента, де виконується цей оператор, і такту, у якому записується в регістр результат цього оператора. Отже, вектори $\mathbf{K}_i$ являють собою теги, що кодують властивості вершин ГСПД, і такий граф називається *просторовим ГСПД*.

Просторовий ГСПД розщеплюється на *просторову конфігурацію* $K_{GS} = (K_S, D_S, A)$ і *конфігурацію подій* $K_{GT} = (K_T, D_T, A)$, яким відповідають структура обчислювача та розклад виконання операторів. При цьому вектори $\mathbf{K}_i = (k_i, s_i, t_i)^{\mathrm{T}}$ розкладаються на вектори $\mathbf{K}_{S_i} = (k_i, s_i)^{\mathrm{T}}$, що відповідають координатам ПЕ, і вектори $\mathbf{K}_{T_i} = t_i$, які означають моменти виконання відповідних операторів у ПЕ $\mathbf{K}_{S_i}$. Тоді часова складова $\mathbf{D}_{T_j} = t_j$ вектора залежності $\mathbf{D}_j$ дорівнює затримці $t_j$ пересилання або оброблення відповідної змінної. Розклад виконання алгоритму — конфігурація подій, а його пошук — знаходження значень $\mathbf{K}_{T_i}$.

Можна вважати, що матриця $K$ кодує деякий допустимий розв'язок, оскільки матриця $D$ обчислюється за рівнянням

$$D = KA. \tag{2}$$

Пошук оптимального структурного рішення полягає у знаходженні такої матриці $K$, яка мінімізує заданий критерій якості, наприклад, такий, як у праці [21]. Для цього спочатку задаються координати векторів матриці $D_O$, що забезпечують умови мінімального значення $T_C$, а координати векторів $\mathbf{K}_i$ знаходяться із співвідношення

$$\mathbf{K} = D_O A_O^{-1}, \tag{3}$$

де $D_O$ — матриця векторів-дуг; $A_O$ — матриця інцидентності кістяка ГСПД.

Для пошуку ефективних структурних рішень необхідно керуватися такими закономірностями.

Просторовий ГСПД є *коректним*, якщо в матриці $K$ немає двох однакових векторів, тобто

$$\forall \mathbf{K}_i, \mathbf{K}_j \ \ (\mathbf{K}_i \neq \mathbf{K}_j, \ i \neq j). \tag{4}$$

Розклад виконання алгоритму з тривалістю ітерації в $L$ тактів є *коректним*, якщо оператори, які відображаються в один і той самий ПЕ, виконуються в різних тактах, тобто

$$\forall \mathbf{K}_i, \mathbf{K}_j \ (k_i = k_j, s_i = s_j) \Rightarrow t_i \not\equiv t_j \bmod L. \tag{5}$$

Причому наступний оператор виконується не раніше за попередній, тобто

$$\forall \mathbf{D}_j \neq \mathbf{D}_{D_j} \ (t_i \geq 0), \tag{6}$$

де $\mathbf{D}_{D_j}$ — вектор-дуга межітераційної залежності $\mathbf{D}_{D_j} = (k_i, s_i, -wL)^{\mathrm{T}}$, яка означає затримку на *w* циклів (ітерацій).

Однотипні оператори необхідно відображати в ПЕ одного й того самого типу, тобто

$$\mathbf{K}_i, \mathbf{K}_j \in K_{p,q} (k_i = k_j = p, s_i = s_j = q), \ \left| K_{p,q} \right| \leq L, \tag{7}$$

де $K_{p,q}$ — множина векторів-вершин операторів *p*-го типу, що відображаються в *q*-й ПЕ *p*-го типу $(q = 1,2,\ldots, q^p_{\max})$.

Дорівнювати нулю має також сума векторів-дуг $\mathbf{D}_j$, що входять у будь-який із циклів графу, включаючи дуги межітераційної залежності $\mathbf{D}_{D_j}$. Тоді для *i*-го циклу

$$\sum_j b_{i,j} \mathbf{D}_j = (0,0,0)^{\mathrm{T}}, \tag{8}$$

де $b_{ij}$ — елемент *i*-го рядка цикломатичної матриці ГСПД. Дуги межітераційної залежності $\mathbf{D}_{D_j} = (k_i, s_i, -wL)^{\mathrm{T}}$ означають затримку на *w* циклів (ітерацій).

Отже, просторовий ГСПД являє собою об'єднання ациклічного графу, який виконує обчислення однієї ітерації, і множини дуг $\mathbf{D}_{D_j}$, які означають межітераційну затримку змінних на $wL$ тактів.

У простому випадку кожен оператор алгоритму виконується за один такт. Такий випадок є натуральним у проектуванні конвеєрних обчислювальних пристроїв на рівні регістрових передач. Складні оператори можуть виконуватись за кілька тактів. Але, працюючи у конвеєрному режимі, відповідні їм ПЕ мають вигляд ланцюжка конвеєрних ступенів, кожний з яких виконує однотактову затримку.

За цих умов пошук розкладу алгоритму полягає у такому. Необхідно призначити всім векторам-дугам $\mathbf{D}_i \in D_O$ координату $t_i = 1$, тобто встановити затримку в один такт і знайти $K_T$ з відношення (3). Решту елементів матриці $D_T$ знаходять з рівнянь (2) і (8). Якщо для деяких з них не справджується нерівність (6), то збільшують координату $t_i$ певних векторів $\mathbf{D}_i \in D_O$ і повторюють пошук.

Решту координат векторів $\mathbf{K}_i$ знаходять з умов (4)–(8). Таким чином, буде побудовано найбільш швидкий розклад, оскільки кожен оператор виконується за один такт і немає зайвих затримок. Якщо побудована матриця $K$ не задовольняє критерій оптимальності, наприклад якихось ресурсів необхідно надто багато, то змінюють деякі вектори $\mathbf{D}_i$, які пов'язані з цими ресурсами або змінюють $L$ і повторюють пошук розкладу.

Цілочислова оптимізація розкладу полягає в побудові ряду оптимізованих рішень та виборі найбільш переважного з них за заданим критерієм оптимальності. Кожне з таких рішень взаємно однозначно кодується матрицею $K$. Початкове оптимізоване рішення можна обчислити як розклад, отриманий за правилами, описаними вище. Решту оптимізованих рішень знаходять завдяки еквівалентним перетвореннням просторового ГСПД, наприклад, за допомогою локальної перестановки векторів $\mathbf{K}_i$ у просторі та взаємної перестановки таких векторів з урахуванням умов (4)–(8). Таким чином, відбувається сканування простору ефективних рішень. При цьому доречно використовувати відомі методи оптимізації, такі як метод гілок і меж, генетичний метод та ін.

## ДОСКОНАЛИЙ КІСТЯК ГСПД

Під час формування розкладу ГСПД за методом просторового ГСПД під час пошуку решти векторів $\mathbf{D}_{T_j}$ з рівняння (2) дуже ймовірна поява некоректного розкладу за умовою (6). Тому пошук розкладу залишається складною задачею комбінаторної оптимізації. Успішне складання розкладу за цим методом істотно залежить від вибору кістяка ГСПД. Щоб вибрати кістяк, який би забезпечував швидкий пошук оптимального розкладу, розглянемо фактори, що викликають появу некоректного розкладу.

По-перше, у графі може бути *стягувальна дуга*, тобто така, що замикає маршрут з кількох інших дуг, як наприклад, $\mathbf{D}_4 = \mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3$.. При цьому, якщо стягувальна дуга входить у кістяк, можлива неприпустима ситуація, коли часові складові векторів-дуг, які з'єднує ця дуга, можуть бути від'ємними. Приклад такого випадку показано на рис.2, *а*, з якого видно, що якщо взяти вісь $ot$ за вісь часу, то часова складова вектора $\mathbf{D}_3$ виявиться від'ємною.

По-друге, у графі є *поперечні дуги*, тобто такі дуги, які не є стягувальними, але додавання яких до кістяка утворює у графі контури, тобто вони не входять у кістяк за визначенням [23]. На рис. 2, *б* дуги $\mathbf{D}_i$, $\mathbf{D}_k$ є альтернативними поперечними дугами. При цьому дуга $\mathbf{D}_i$ входить до довшого марш-

руту. Якщо $\mathbf{D}_i$ не входить до кістяка ГСПД, то після визначення розкладу за рівнянням (3) з виразу (2) випливає, що $\mathbf{D}_{T_i} < 0$, тобто такий розклад буде хибним. Отже, потрібно зважати на поперечні дуги, що входять у коротші маршрути, і вилучати їх з графу під час побудови кістяка.

Розглянемо такі поперечні дуги, наявність яких може призвести до хибного розкладу. Таку дугу назвемо *критичною поперечною дугою*. У графі можуть бути рівнозначні маршрути з однієї вершини в іншу, наприклад, як на рис. 2, *в*, де $\mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3 = \mathbf{D}_4 + \mathbf{D}_5 + \mathbf{D}_6$. Тут критичною поперечною дугою є така, що замикає рівнозначний маршрут в контур, тобто $\mathbf{D}_3$ або $\mathbf{D}_6$. Також критичною поперечною дугою є така дуга, яка належить до контуру, котрий формує підграф з декількома вершинами-витоками. Типовий такий підграф зображено на рис. 1, *г*, для якого критичними дугами є всі дуги, які входять у стоки підграфу, тобто $\mathbf{D}_1, \ldots, \mathbf{D}_4$.

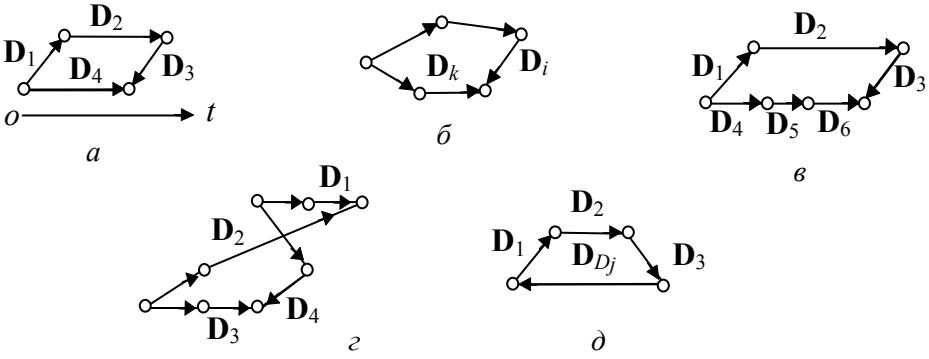


*Рис. 2*. Приклади підграфів для побудови кістяка

В обох випадках розглянуті підграфи належать до класу узгоджених графів. *Узгодженим графом* називають граф, у кожному циклі якого кількість дуг є парною, а сума цих векторів-дуг дорівнює нулю [20]. У кістяк необхідно включити повністю хоча б один з рівнозначних маршрутів. Тому, якщо в початковому графі відмітити всі критичні поперечні дуги, то частина їх обов'язково потрапить у результуючий кістяк, що може призвести до хибного розкладу. Тоді, щоб отримати коректний розклад, необхідно в кістяку відмітити вершини, у які заходять критичні поперечні дуги, і в процесі пошуку розкладу перевіряти і корегувати часові складові дуг, які заходять у відмічені вершини.

Критичні поперечні дуги ускладнюють задачу складання розкладу для ГСПД. Якщо у ГСПД немає таких дуг, то складність пошуку розкладу оцінюється трудністю обчислення формул (2) і (3), тобто як $O(n^2)$. Кожна додаткова критична поперечна дуга ускладнює цю задачу в середньому в $L/2$ разів, тобто за великої кількості таких дуг задача перетворюється у NP-повну. Для виконання комбінаторної оптимізації розкладу слід відмітити критичні поперечні дуги або вершини, у які вони входять.

По-третє, якщо алгоритм є циклічним, то цикли в ньому замикаються дугами міжітераційних залежностей $\mathbf{D}_{D_j}$, причому, наприклад, для графу, показаному на рис. 2, *д*, $\mathbf{D}_{D_j} = -(\mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3)$ і $\mathbf{D}_{DT_i} = -L$. Тобто для та-

кої дуги часова складова відома заздалегідь і вона не може бути змінена. Оскільки ця дуга замикає цикли, недоцільно залишати її у кістяку ГСПД.

Наведені фактори необхідно враховувати під час складання розкладу. Щоб таких урахувань було якомога менше, доцільно використовувати такий кістяк ГСПД, який спеціально підібраний для знаходження розкладу. Назвемо такий кістяк *досконалим*. Таким чином, можна сформулювати таке визначення.

*Досконалий кістяк* — це кістяк ГСПД, у якому: а) немає стягувальних дуг, б) немає дуг міжітераційних залежностей, в) дуги входять у найдовші маршрути, г) відмічені вершини, у які заходять критичні поперечні дуги.

У праці [23] запропоновано алгоритм побудови досконалого кістяка ГСПД і доведено такі твердження.

**Твердження 1.** До досконалого кістяка належить критичний шлях алгоритму.

**Твердження 2.** Розклад, побудований на основі досконалого кістяка і рівняння (3), належить до розкладу, складеного за принципом найшвидшого призначення операторам моментів виконання (ASAP).

## ПРИКЛАД ПОШУКУ РОЗКЛАДУ ГСПД

Щоб оцінити дієвість методу, розглянемо пошук розкладу для ГСПД (див. рис. 1). У досконалому кістяку цього ГСПД (рис. 3) вектор-дуга $\mathbf{D}_B$ з'єднує початок системи координат з довільною вершиною. Вона штучно додана для того, щоб кількість дуг у кістяку збігалась з кількістю вершин. Це є умовою того, що рівняння (3) є невиродженим.

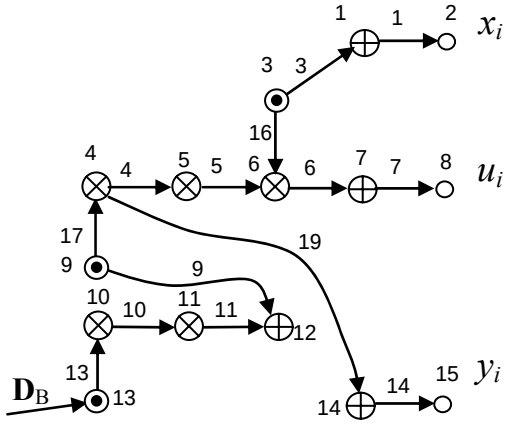


*Рис. 3*. Досконалий кістяк для ГСПД (див. рис. 1)

У цьому прикладі, як у тестовому, використовуються суматори, що мають затримку в один такт, та конвеєрні блоки множення із затримкою у два такти. З урахуванням цього складається матриця затримок дуг кістяка:

$$D_{OT} = \begin{matrix}(1 & x_1 & 2 & 2 & 2 & 1 & x_2 & 2 & 2 & x_3 & 1 & x_4 & x_5 & x_6 & 0)^{\mathrm{T}},\\ {\scriptstyle 1} & {\scriptstyle 3} & {\scriptstyle 4} & {\scriptstyle 5} & {\scriptstyle 6} & {\scriptstyle 7} & {\scriptstyle 9} & {\scriptstyle 10} & {\scriptstyle 11} & {\scriptstyle 13} & {\scriptstyle 14} & {\scriptstyle 16} & {\scriptstyle 17} & {\scriptstyle 18} & {\scriptstyle B}\end{matrix}$$

де $x_1,\ldots,x_5$ — невідомі величини, які залежать від векторів з рівнянь циклів (8), кількість яких дорівнює кількості дуг, які доповнюють кістяк до ГСПД. Тобто кожен цикл графу повинен мати принаймні одну змінну. У цьому ГСПД п'ять циклів і відповідно система складається з п'яти рівнянь:

$$\begin{cases} \mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3 = 0, \\ \mathbf{D}_4 + \mathbf{D}_5 + \mathbf{D}_6 + \mathbf{D}_7 + \mathbf{D}_8 + \mathbf{D}_{17} = 0, \\ \mathbf{D}_{14} + \mathbf{D}_{15} + \mathbf{D}_{18} = 0, \\ \mathbf{D}_4 + \mathbf{D}_5 + \mathbf{D}_6 - \mathbf{D}_{12} - \mathbf{D}_9 + \mathbf{D}_{17} = 0, \\ \mathbf{D}_{10} + \mathbf{D}_{11} + \mathbf{D}_{13} - \mathbf{D}_9 + \mathbf{D}_{17} + \mathbf{D}_{19} - \mathbf{D}_{18} = 0. \end{cases} \quad (9)$$

Розв'язавши систему рівнянь (9), знаходимо $x_1 = 6;\ \ x_2 = 5;\ \ x_3 = 5;$ $x_5 = 0;\ \ x_6 = 6$. Змінна $x_4 = 6$ вибирається довільно. Період алгоритму $L = 7$ вибирається як довжина критичного шляху:

$$\mathbf{D}_8 = -(\mathbf{D}_4 + \mathbf{D}_5 + \mathbf{D}_6 + \mathbf{D}_7 + \mathbf{D}_{17}).$$

За рівнянням (2) знаходимо матрицю $K_T$:

$$K_T = A_O \begin{matrix} (1 & 3 & 4 & 5 & 6 & 7 & 9 & 10 & 11 & 13 & 14 & 16 & 17 & 18 & B \end{matrix}$$
$$\begin{pmatrix} 1 & 6 & 2 & 2 & 2 & 1 & 5 & 2 & 2 & 5 & 1 & 4 & 0 & 6 & 0 \end{pmatrix}^T =$$
$$= \begin{pmatrix} 8 & 9 & 2 & 4 & 6 & 8 & 10 & 11 & 4 & 5 & 7 & 9 & 0 & 6 & 7 \end{pmatrix}^T.$$

Перевіряємо справедливість вимоги (5).

Решту координат матриці *K* отримуємо згідно з вимогою (4), беручи до уваги, що кількість вершин, розміщених в одному рядку, паралельному осі *ox,* для мінімізації апаратних витрат має прямувати до *L*:

$$\begin{matrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix}$$
$$K = \begin{pmatrix} 2 & 3 & 4 & 1 & 1 & 1 & 2 & 3 & 4 & 1 & 1 & 2 & 4 & 2 & 3 \\ 1 & 4 & 3 & 2 & 2 & 2 & 1 & 4 & 3 & 2 & 2 & 1 & 3 & 1 & 4 \\ 8 & 9 & 2 & 4 & 6 & 8 & 10 & 11 & 4 & 5 & 7 & 9 & 0 & 6 & 7 \end{pmatrix}.$$

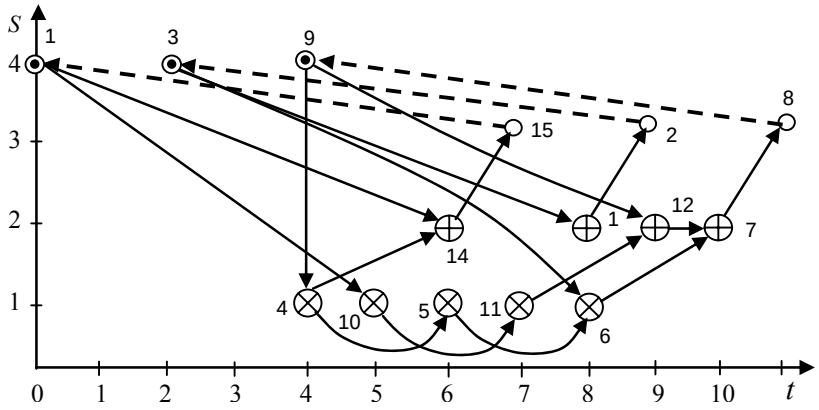


*Рис. 4*. Просторовий ГСПД, який обчислює рівняння (1)

Графічне зображення побудованого просторового ГСПД показано на рис. 4. За цим графом формальним чином за методикою, викладеною

у праці [24], можна скласти опис обчислювального пристрою мовою VHDL. Структуру відповідного пристрою показано на рис. 5.

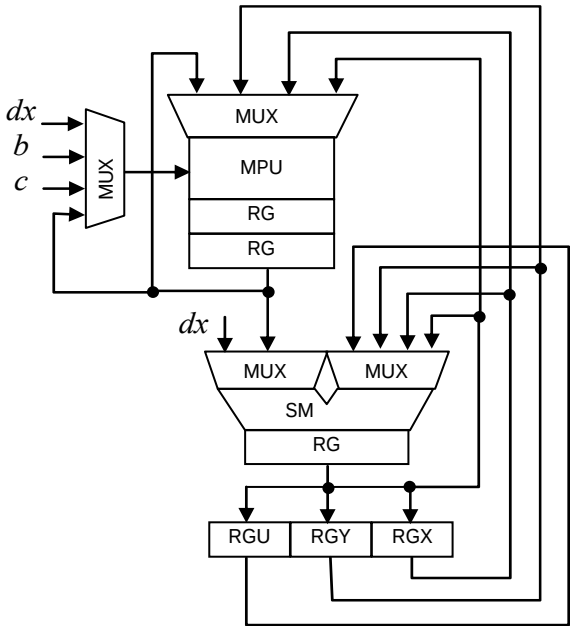


*Рис. 5*. Структура пристрою, що відповідає просторовому ГСПД (див. рис. 4)

Іншим тестовим прикладом є синтез хвильового рекурсивного фільтра п'ятого порядку [6]. У таблиці наведено результати такого синтезу з використанням простого та конвеєрного блоків множення із затримкою на два такти. Їх порівняння з відомими результатами, отриманими за іншими методами, показує, що запропонований метод дає мінімальні апаратні витрати у кількості блоків множення, суматорів та регістрів за рахунок збільшеної кількості мультиплексорів.

Результати синтезу обчислювального пристрою для обчислення еліптичного фільтра п'ятого порядку

| **Параметр** | **Просторовий ГСПД, блок множення** | | | **SPAID [25], блок множення** | | | **HAL[6], блок множення** | | |
|---|---|---|---|---|---|---|---|---|---|
| | простий | конвеєрний | | простий | конвеєрний | | простий | конвеєрний | |
| Блоки множення | 2 | 1 | 1 | 3 | 2 | 2 | 3 | 2 | 1 |
| Суматори | 3 | 3 | 2 | 3 | 3 | 2 | 3 | 3 | 2 |
| Входи мультиплексорів | 55 | 41 | 36 | 37 | 35 | 24 | 36 | 37 | 28 |
| Регістри | 11 | 11 | 10 | 21 | 21 | 21 | 12 | 12 | 12 |
| Період обчислень $L$ | 17 | 17 | 19 | 17 | 17 | 19 | 17 | 17 | 19 |

Цей метод був застосований для синтезу конвеєрних процесорів швидкого перетворення Фур'є, дискретного косинусного перетворення, рекурсивних фільтрів. Деякі з них як віртуальні модулі, що описані мовою Verilog,

зберігаються у спільноті `opencores.org` і є доступними для порівняння, як наприклад, рекурсивний фільтр високого порядку, що динамічно перебудовується [26].

## ВИСНОВКИ

У роботі запропоновано формальний метод складання розкладу для графу синхронних потоків даних, який забезпечує синтез конвеєрних обчислювальних пристроїв з регламентовано великою пропускною здатністю та мінімізованими апаратними витратами.

Приклад розроблення пристрою для розв'язання диференціального рівняння свідчить, що для невеликих графів можливе знаходження точного розв'язку задачі синтезу. Досвід розроблення більш складних проектів, таких як процесори швидкого перетворення Фур'є, рекурсивні фільтри високого порядку, показує, що метод варто застосовувати з комбінаторною оптимізацією. Але така оптимізація менш складна, ніж з використанням інших методів через велику кількість обмежень, які накладаються на просторовий граф синхронних потоків даних.

Метод може бути застосований не тільки для розроблення конвеєрних спеціальних процесорів, але і для програмування паралельних мультипроцесорних систем.

## ЛІТЕРАТУРА


1. *Сергиенко А.М.* Алгоритмические модели обработки потоков данных / А.М. Сергиенко, В.П. Симоненко // Электронное моделирование. — 2008. — Т. 30. — № 6. — С. 49–60.
2. *Lee E.A.* Synchronous Datafow / E.A. Lee, D.G. Messerschmitt // Proc. IEEE. — 1987. — V. 75. — N 9. — P. 1235–1245.
3. *Edwards S.* Design of Embedded Systems: Formal Models, Validation, and Synthesis / S. Edwards, L. Lavagno, E.A. Lee, A. Sangiovanny-Vincentelli // Proc. IEEE. — 1997. — V. 85. — N 3. — P. 366–390.
4. *Lee E.A.* Static scheduling of synchronous data flow programs for digital signal processing / E.A. Lee, D.G. Messerschmitt // IEEE Trans. on Computers. — 1987. — V. 36. — N 1. — P. 24–35.
5. *O'Neil T.W.* Retiming synchronous data-flow graphs to reduce execution time / T.W. O'Neil, E.H.M. Sha // IEEE Trans. on Signal Processing. — 2001. — **49**, N 10. — P.2397–2407.
6. *Paulin P. G.* HAL: A multi-paradigm approach to automatic data path synthesis / P.G. Paulin, J.P. Knight, E.F. Girczyc // Proc. 23-rd IEEE Design Automation Conf, Las Vegas, NV, July 1986. — 1986. — P. 263–270.
7. *Chao L.* Rotation scheduling: A loop pipelining algorithm / L. Chao, A. LaPaugh, E. Sha // Proc 30-th Design Automation Conf., DAC'93, June 1993. — 1993. — P. 566–572.
8. *The Systhesis* Approach to Digital System Design / Editors P. Micheli, U.Lauther, P. Duzy. — Kluwer Academic Pub, 1992. — 415 p.
9. *ЭВМ и теория расписаний* / Под ред. И.Г. Кофмана. — М.: Мир, 1979. — 460 с.
10. *Paulin P.G.* Force – Directed Sheduling for the Behavioral Synthesis of ASICs / P.G. Paulin, J.P. Knight // IEEE Trans. CAD. —1988. — **7**, N 3. — P. 356–370.

11. *Park N.* Module Assignment and Interconnect Sharing in Register-Transfer Synthesis of Pipelined Data Paths / N. Park, F.J. Kurdahi // Proc. IEEE Int. Conf. on Computer Aided Design. — Santa Clara, Calif. — 1989. — P. 16–19.
12. *Hwang K. S.* Scheduling and hardware sharing in pipelined data paths / K.S. Hwang, A.E. Casavant, C.T. Chang, M.A. d'Abreu // Proc. Int'l Conf. on Computer Aided Design. — 1989. — P. 24–27.
13. *Catthoor F.* Application-specific architectural methodologies for high-throughput digital signal and image processing / F. Catthoor, H. De Man // IEEE Trans. on Acoustics, Speech and Signal Processing. — 1990. — **38**, N 2. — P. 339–349.
14. *Tseng C.* Automated Synthesis of Data Paths in Digital Systems / C. Tseng, D. Siewiorek // IEEE Trans. on Computer-Aided Design. — 1986. — N 7. — P. 379–395.
15. *Kurrdahi F.J.* REAL: A Program for Register Allocation / F.J. Kurrdahi, A.C. Parker // Proc. 24-th ACM/ IEEE Design Automation Conf. DAC-87. — 1987. — P. 210–215.
16. *Tucker A.* Coloring a family of circular arcs / A. Tucker // SIAM J. Appl. Math. — 1975. — **29**, N 3. — P. 493–502.
17. *Springer D.L.* Exploiting the Special Structure of Conflict and Compatibility Graphs in High-Level Synthesis / D.L. Springer, D.E. Thomas // Proc. Int. Conf. Computer Aided Design (ICCAD). — 1990. — P. 254–257.
18. *Robert Y.* Introduction to Scheduling / Y. Robert, F. Vivien – Ed-s. CRC Press, Taylor and Francis Group. — 2010. — 310 p.
19. *Lee E.A.* Static scheduling of synchronous data flow programs for digital signal processing / E.A. Lee, D.G. Messerschmitt // IEEE Trans. on Computers. — 1987. — **36**, N 1. — P. 24–35.
20. *Воеводин В.В.* Математические модели и методы в параллельных процесах / В.В. Воеводин. — М: Наука. — 1986. — 296 с.
21. *Сергиенко А.М.* Отображение периодических алгоритмов в программируемые логические интегральные схемы / А.М. Сергиенко, В.П. Симоненко // Электронное моделирование. — 2007. — **29**, № 2. — С. 49–61.
22. *Евстигнеев В.А.* Применение теории графов в программировании / В.А. Евстигнеев; под ред. А.П. Ершова. — М: Наука, 1985. — 352 с.
23. *Сергієнко А.М.* Досконалий кістяк графе алгоритму / А.М. Сергієнко // Інформатика і обчислювальна техніка: зб. наук. праць. — 2007. — № 46. — С. 62–67.
24. *Сергиенко А.М.* VHDL для проектирования вычислительных устройств / А.М. Сергиенко. — К.: ДиаСофт, 2003. — 210 с.
25. *Haroun B.S.* SPAID: An Architectural Synthesis Tool for DSP Custom Applications / B.S. Haroun, M.I. Elmasry // Proc. IEEE Custom Integrated Circuits Conf. — Rochester, N.Y. — May 16–19. — 1988. — P. 14.4/1–14.4/5.
26. *Sergiyenko A.* Low-Pass IIR Filter / A. Sergiyenko, O. Uzenkov. — 2010. — Available at http://opencores.org/project,lp_iir_filter

*Надійшла 25.08.2015*