doi: https://doi.org/10.15407/emodel.42.02.025
УДК 004.383

А.М. Сергієнко, д-р техн. наук,
В.О. Романкевич, д-р техн. наук, **А.А. Сергієнко**
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
(Україна, 03056, пр-т Перемоги, 37,
тел. +38(068) 8123376, e-mail: aser@comsys.kpi.ua;
тел. +38(096) 1607909, e-mail: romankev@scs.ntu-kpi.kiev.ua;
тел. +38(096) 8127583, e-mail: a.serhienko@comsys.kpi.ua)

Генетичне програмування спеціалізованих конвеєрних пристроїв

Запропоновано метод синтезу спеціалізованих конвеєрних пристроїв, який ґрунтується на генетичному програмуванні. Метод полягає у відтворенні алгоритму просторовим графом синхронних потоків даних, кодуванні його матриці вершин-операторів хромосомою та використанні алгоритму генетичної оптимізації. Високу ефективність методу показано на прикладі синтезу процесора для дискретного косинусного перетворення, конфігурованого у програмованій логічній інтегральній схемі.

К л ю ч о в і с л о в а: програмовані логічні інтегральні схеми, граф потоків даних, генетичне програмування.

Проектування на системному рівні, з одного боку, необхідне для підтримання належної високої продуктивності розробників мікросхем та проектів для програмованих логічних інтегральних схем (ПЛІС), з іншого боку, воно менше піддається автоматизації, ніж логічне чи фізичне проектування. Тому тепер таке проектування стало центральною проблемою для розробників САПР мікросхем [1].

Напрямок поведінкового системного синтезу ґрунтується на відображенні алгориму, що задає поведінку пристрою, у шукану структуру процесора. Таке відображення найчастіше виконується за допомогою складання розкладу алгоритму, поданого у вигляді графа потоків даних (ГПД) для вибраної множини ресурсів. При виконанні поведінкового синтезу такими «жадібними» алгоритмами, як списковe чи силове планування [2], одержують локальний оптимум, який може відповідати пристрою неналежної якості. Часто використовують метод цілочисельного

лінійного програмування, який дозволяє одержати оптимальне рішення через розв'язок системи нерівностей та поєднання етапів вибору ресурсів і складання розкладу [3]. Але цей метод застосовується лише для ГПД з кількістю вершин не більше двох-трьох десятків і його ефективність залежить від точності прийнятого критерію оптимальності.

Використання еволюційних, зокрема генетичних, алгоритмів у поведінковому синтезі дає змогу синтезувати пристрої за ГПД будь-якої складності, автоматизуючи як складання розкладу, так і вибір ресурсів та призначення операцій і змінних на ресурси [2,4]. Оскільки результатом такого проектування є алгоритм, реалізований апаратно, воно одержало назву генетичного програмування (ГП) [5].

У роботах [6,7] описано метод синтезу конвеєрних пристроїв для ПЛІС на основі просторового графа синхронних потоків даних (ГСПД) і показано його дієвість на прикладах нескладних проектів. Для виконання проектування пристроїв будь-якої складності розглянемо метод ГП, що ґрунтується на перетвореннях просторового ГСПД.

Методи ГП. Еволюційні алгоритми абстрагуються від своїх біологічних аналогів, а також вносять зміну до семантики блага еволюції [4]. Простір пошуку $\mathbb{G}$ в еволюційних алгоритмах означає множину усіх можливих хромосом, тобто рядків $g \in \mathbb{G}$, що кодують конкретні рішення. Рядок-хромосома g кодує певний генотип конкретної особини x, яка є кандидатом на рішення з простору рішень $\mathbb{X}$. Особина, індивід чи фенотип x, одержується за відображенням хромосоми $x = \text{Phenotype}(g)$ або як її модель $p.x$. Рівень відповідності оптимальному рішенню визначається за функцією оптимальності $v(x)$, яка рухає процес еволюції у необхідному напрямку і традиційно в еволюційних алгоритмах називається функцією придатності.

Базовий підхід, який використовується в еволюційних алгоритмах, пояснюється наступним алгоритмом.

А л г о р и т м 1.

Вхідні дані: TCriterion() — критерій зупинки алгоритму;
Cmpf — відношення порядку між особинами в популяції;
ps — розмір популяції.

Змінні: *t* — лічильник поколінь;
Pop — популяція, яка є множиною особин *p.x*;
Popm — частина популяції для розмноження;
v — величина придатності.

Результат — найкраще рішення.

```
begin
    t = 0;
    Pop = Initialize(ps);
    while not TCriterion() {
```

```
        t = t + 1;
        v = Fitness(Pop(t–1), Cmpf),
        Popm = Select(Pop(t–1), v, ps);
        Pop(t) = Reproduce(Popm);
        }
    return Phenotype(Optimal(Pop));
end;
```

Функція Initialize(*ps*) генерує початкову випадкову популяцію з номером $t = 0$, яка складається з *ps* хромосом *g*. Функція TCriterion() перевіряє, чи справджується критерій закінчення алгоритму. Значення придатності *v* визначається у більшості алгоритмів для хромосом *g* кожної особини через обчислення критерію ефективності та порівняння за ним з іншими особинами з множини *Pop*(*t*). При цьому за хромосомою *g* будується спрощена модель фенотипу *p.x*, для якої, власне, дається оцінка ефективності Fitness() з урахуванням відношення *Cmpf*. Функція Fitness() повинна виділяти найкращі особини за умови збереження різноманіття особин, щоби конвергенція алгоритму не призвела до єдиного стійкого локального оптимуму.

Функція відбору Select(*Pop*(t–1), *v*, *ps*) вибирає множину *ps* особин *Popm*, які варті розмноження. Функція Reproduce(*Popm*), тобто функція відтворення чи розмноження генерує нову популяцію з використанням операцій створення особини, її мутації, схрещування тощо. Функції Optimal(*Pop*) та Phenotype використовуються для виділення хромосом найліпших особин та побудови їх фенотипів.

У ГП бітовий вектор використовується як хромосома $g \in \mathbb{G}$. Генетичне програмування базується на еволюційних методах створення нових алгоритмів та програм і найчастіше оперує ГПД, який має деревовидну структуру. Відповідна хромосома кодує певне дерево [8].

Як правило, ГП у різних методах застосовується лише на окремих стадіях синтезу: виборі ресурсів, складанні розкладу чи призначенні на ресурси. Так, у системі SPARCS розклад алгоритму відшукується за допомогою лінійного програмування, а розподіл операцій по ресурсах ПЛІС — за допомогою ГП [9]. У роботі [10] ГП використовується для підвищення ефективності спискового планування, а в роботі [11] — для розпаралелювання циклів. У роботі [12] за допомогою ГП виконується розклад періодичного алгоритму на заданій множині ресурсів. При цьому за жадібним алгоритмом отримано початкове рішення, а остаточне — генетичним алгоритмом без мутацій.

У роботі [13] ГП використано для складання розкладу з мінімізацією апаратних витрат та затримки, а в [14] — для призначення на ресурси та пошуку їх з'єднання. У [15] описано ієрархічний алгоритм ГП, в якому

спочатку виконано вибір ресурсів, а потім — складання розкладу. Лише деякі алгоритми ГП, наприклад описаний в [16], при оптимізації враховують апаратні витрати на міжз'єднання та мультиплексори, притому тільки наприкінці оптимізації.

У різних методах використовують різні способи кодування хромосом. У роботі [17] запропоновано кодувати хромосому як список пріоритетів, що визначає, у якому порядку слід обирати операції за алгоритмом спискового планування. У деяких роботах [14, 18, 19] запроваджено кодування хромосоми на основі пар операція—ресурс, кожна з яких задає відношення операції та ресурсу, де вона виконується. В роботі [20] описано метод NSGA-II, за яким ген задає відношення операція—ресурс і на основі графа алгоритму поетапно виконується побудова структури, складання розкладу, призначення на регістри.

Є багато методів ГП, що ґрунтуються на просторовому відтворенні графа алгоритму. Так, у методі паралельного розподіленого ГП (PDGP) деревовидний ГПД подано у n-вимірній решітці. Вершини графа відображають операції, а еволюції підлягають мітки вершин та зв'язки між вершинами, за винятком міток з обмеженнями. Крім того, зв'язкам відповідають вагові значення, які також еволюціонують. Цим методом синтезуються штучні нейронні мережі [21].

Для синтезу систем з обмеженим паралелізмом використовується метод прикладного декартового ГП (ECGP), у якому вершини графа подано у двовимірній решітці, при цьому виконання операторів вершин, що стоять лівіше, передує виконанню тих, що стоять правіше. Хромосома зображена як множина nm генів, які кодують вершини решітки розміром nm. Гени кодують зв'язки між вершинами та їхні функції. Додано їхні нові функції розмноження, наприклад компресію, за якою кілька вершин замінюються вершиною-модулем, розкриття, що замінює вершину-модуль відповідною множиною вершин, та мутацію вершин-модулів [22]. Цей метод нагадує методи згортання (folding) та розгортання (unfolding) ГСПД [23], але він не стосується складання розкладу, яке повинне виконуватись окремо.

Метод декартового ГП має багато різних модифікацій. Так, в [24] досліджено алгоритм ECGP, який формує популяцію з елітних батьків та дітей у пропорції $1 : \lambda$, при цьому виконується лише мутація в окремих генах без схрещування. У [25] розглянуто метод ГП з геометричною семантикою (GSGP), в якому додано геометричний семантичний оператор, що спрощує обчислення функції придатності.

Найбільш складною операцією в алгоритмах ГП є обчислення функції придатності Fitness(), яку можна визначити лише за результуючою структурою пристрою та за одержаним розкладом алгоритму. Придат-

ність також може включати інформацію про належність особини до певної підмножини особин з аналогічними властивостями — ніші. Таким чином, враховується не тільки якість кандидата на рішення, але й якість різноманітності популяції, що може покращити шанс визначити глобальний оптимум [26].

Серед усіх методів є перспективними методи ECGP [22], в яких хромосоми кодують просторове положення операторних вершин ГПД, бо таке кодування дає змогу ефективно знаходити функції придатності. Зокрема, тривалість виконання алгоритму визначається як горизонтальні розміри області поширення вершин ГПД.

Методи ГП також розрізняються за різними функціями Select(), за якими обираються особини для нової популяції. Згідно з алгоритмом пропорційного відбору ймовірність P (p_i) того, що окрема особина $p_i \in Popm$ потрапить у множину Pop, пропорційна параметру v ($p_i.x$) [27]. Серед кількох різновидів цього алгоритму найпоширенішим є метод Монте-Карло, який полягає в упорядкуванні особин на моделі колеса рулетки відповідно до їхньої придатності v та випадковому виборі особин послідовно за правилами гри в рулетку [28]. Алгоритм турнірного відбору створює множину найкращих особин популяції та не відбирає майже жодної поганої особини [29].

Отже, описані вище методи ГП пристосовані лише для виконання окремих кроків структурного синтезу обчислювальних пристроїв або ці кроки виконуються послідовно, через що оптимізація часто є недосконалою. Крім того, у більшості методів результатом синтезу є пристрій, налаштований на одноразове виконання алгоритму, а не виконання алгоритму в конвеєрному режимі. Якщо синтезується конвеєрний пристрій, то він виконує алгоритм з періодом один такт.

Генетичне програмування просторового ГСПД. Метод проектування конвеєрних обчислювачів, описаний в [6,30], полягає в поданні ГСПД алгоритму в тривимірному просторі у вигляді конфігурації алгоритму $K_G = (K, D, A)$ та у розщепленні її на просторову конфігурацію $K_{GS} = (K_S, D_S, A)$ і конфігурацію подій $K_{GT} = (K_T, D_T, A)$. Тут K — матриця векторів-вершин $\mathbf{K}_i$, що відповідають операторам алгоритму; D — матриця векторів-дуг $\mathbf{D}_j$, які задають безпосередні інформаційні зв'язки між операторами; A — матриця інцидентності ГСПД. У векторі-вершині $\mathbf{K}_i = (k_i, s_i, t_i)^T$ координати k_i, s_i, t_i відповідають типу оператора, номеру процесорного елемента (ПЕ), де виконується даний оператор, і такту, в якому записується у регістр результат цього оператора. За просторовою конфігурацією знаходять шукану структуру пристрою, а за конфігурацією подій — розклад виконання операторів.

Для просторового ГСПД властиво те, що матриця K кодує деяке допустиме рішення. Тому при синтезі конвеєрного пристрою за допомогою просторового ГСПД цілочисельна оптимізація розкладу полягає в побудові ряду оптимізованих рішень та виборі найбільш оптимального для матриці K за заданим критерієм оптимальності. Цей критерій нескладно визначити за матрицями K, D, A. Оптимізовані рішення одержують завдяки еквівалентним перетворенням просторового ГСПД, наприклад, за допомогою локальної перестановки векторів $\mathbf{K}_i$ у просторі та взаємної перестановки таких векторів з урахуванням умов корректності конфігурацій.

Метод просторового ГСПД можна ефективно реалізувати у ГП, бо за своїми властивостями він нагадує елементи методу ECGP. У роботі [6] показано критерій оптимальності, який нескладно перетворити у функцію Fitness. Необхідна для цього матриця D обчислюється за рівнянням $D = KA$. Після корекції матриці D відповідні зміни генів $\mathbf{K}_i$ знаходимо зі співвідношення $K = D_O A_O^{-1}$, де D_O — матриця векторів-дуг; A_O — матриця інцидентності кістяка ГСПД.

Щоб запобігти появі нежиттєздатних особин у популяціях, слід дотримуватися коректності просторового ГСПД за умови

$$\forall\ \mathbf{K}_i, \mathbf{K}_j\ (\mathbf{K}_i \neq \mathbf{K}_j, i \neq j\ ) \tag{1}$$

і коректності розкладу за умов

$$\forall\ \mathbf{K}_i, \mathbf{K}_j\,(k_i = k_j, s_i = s_j\,) \Rightarrow t_i \not\equiv t_j \bmod L, \tag{2}$$

$$\forall \mathbf{D}_j \neq \mathbf{D}_{Dj}\ (t_i \geq 0), \tag{3}$$

$$\sum_j b_{i,j}\,\mathbf{D}_j = (0,0,0)^T, \tag{4}$$

де L — тривалість ітерації (тактів); $\mathbf{D}_{Dj} = (k_j,\ s_j,\ -wL)^T$ — вектор-дуга міжітераційної залежності, яка означає затримку на w циклів (ітерацій); $b_{i,j}$ — елемент i-го рядка цикломатичної матриці ГСПД; $\mathbf{D}_j$ — вектори-дуги, що входять у будь-який цикл графу.

Раціональною умовою побудови структури пристрою є та, що однотипні оператори слід відображати в ПЕ того ж самого типу, тобто

$$\mathbf{K}_i, \mathbf{K}_j \in K_{p,q}\,(k_i = k_j = p, s_i = s_j = q), |K_{p,q}| \leq L, \tag{5}$$

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

Хромосома — це генотип, закодований бінарними кодами як вектор або запис кількох векторних полів, тобто генів. У разі ГСПД матриця K

має всю інформацію, яка повинна міститись у хромосомі. Для традиційного зображення хромосоми слід розгорнути стовпчики матриці K в один рядок g.

Розглянемо приклад тестового алгоритму обчислення диференційного рівняння $y'' + bxy' + cy = 0$ [31]. Воно розв'язується за наступним алгоритмом:

$i = 0;$
while ($x_i < a$) do
  $x_{i+1} = x_i + dx;$
  $u_{i+1} = u_i - b \cdot x_i \cdot u_i \cdot dx - c \cdot y_i \cdot dx;$
  $y_{i+1} = y_i + u_i \cdot dx;$
  $i = i + 1;$
end;

де i — номер ітерації, a, b, c, dx — константи. Після подання алгоритму у вигляді ГСПД та його оптимізації одержано просторовий ГСПД такий, як показано у роботі [7]. Це рішення кодується матрицею

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

і хромосомою

$$g = \{2,1,8; 3,4,9; 4,3,2; 1,2,4; 1,2,6; 1,2,8; 2,1,10; 3,4,11; 4,3,4; 1,2,5; 1,2,7; 2,1,9; 4,3,0; 2,1,6; 3,4,7\}.$$

Тут кожна трійка відтворює окремий ген (k_i, s_i, t_i), при цьому $k_i = 1, 2, 3, 4$ означає відповідно вершини множення, додавання, прийому і передачі змінних між ітераціями. Вершинам з номерами 1, 3, 9 відповідають регістри, які зберігають змінні y_i, x_i, u_i.

У генетичному алгоритмі оптимізації використовується інформація, зібрана до ітерації t для створення кандидатів рішення, що оцінюються на ітерації $t + 1$, тобто для відтворення генотипів. В еволюційних алгоритмах існують чотири основні операції відтворення:

1) створення нового генотипу;

2) дублювання, яке дає кілька копій однієї особини;

3) мутація, що полягає у невеликій випадковій зміні генотипу;

4) рекомбінація (схрещування) двох споріднених генотипів у новий генотип.

1. При створенні генотипу просторового ГСПД, наприклад у першій популяції, його параметри s_i, t_i генів задають випадково, дотримуючись умов, за яких особина буде належати простору рішень $g \in \mathbb{G}$. При цьому особина є життєздатною, якщо виконуються умови (1)—(5).

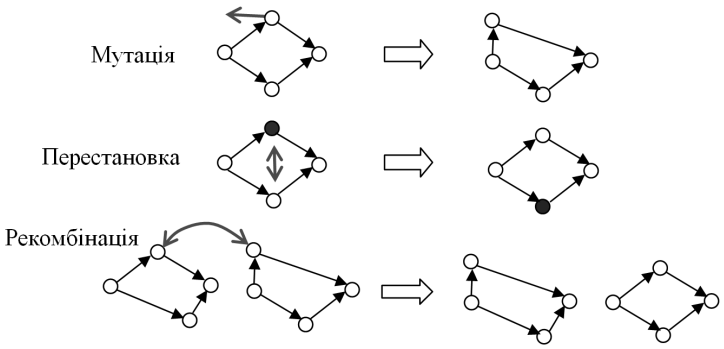

Рис. 1. Операції відтворення для просторового ГСПД

2. Дублювання використовується у випадку ефективного рішення, властивості якого слід використати у наступних популяціях: g_n = duplicate(g), $g \in \mathbb{G}$.

3. Мутація виконується для створення нового життєздатного генотипу через модифікацію наявного і відбувається випадково або детерміновано: g_n = mutate(g), $g \in \mathbb{G}$, $g_n \in \mathbb{G}$. При цьому в довільно вибраному гені (k_i, s_i, t_i) випадково змінюються параметри s_i, t_i таким чином, щоб генотип залишався життєздатним. Аналогічно виконується мутація групи генів (алелей), тобто перестановка. Перестановкою є така мутація, коли обираються два гени одного типу k_i (k_i, s_p, t_p) та (k_i, s_q, t_q) і обмінюються своїми параметрами, даючи гени (k_i, s_q, t_q) та (k_i, s_p, t_p).

4. Рекомбінація, $\mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}$, може траплятись як випадково, так і детерміновано: g_n = recombine(g_a, g_b) : g_a, $g_b \in \mathbb{G} \Rightarrow g_n \in \mathbb{G}$. Для рекомбінації обирають два i-х гени, (k_i, s_{ai}, t_{ai}) та (k_i, s_{bi}, t_{bi}), з двох вибраних випадково генотипів g_a, g_b і параметри першого гена замінюють параметрами другого, отримуючи ген (k_i, s_{bi}, t_{bi}) нового генотипу g_n. Аналогічно виконується рекомбінація алелей.

Операції відтворення наведено на рис. 1. Отже, розмноження — це формування нової сукупності особин через виконання операцій 1—4, які виконуються довільно. Наприклад, мутація може відбуватися після рекомбінації. Але при відтворенні окремі гени, наприклад гени вузлів введення-виведення, що утворюють інтрони, не піддаються змінам, оскільки вони не впливають на характеристики фенотипу.

Найскладнішим завданням алгоритму ГП є реалізація функції Fitness() [32]. При використанні просторового ГСПД ця функція стає набагато простішою, ніж в інших алгоритмах ГП. Наприклад, апаратна складність оцінюється як зважена кількість ПЕ, яка для ПЕ k_i-го типу обчислюється як кількість множин вершин з однаковими координатами k_i та s_i. Враховується також складність входів мультиплексорів [6].

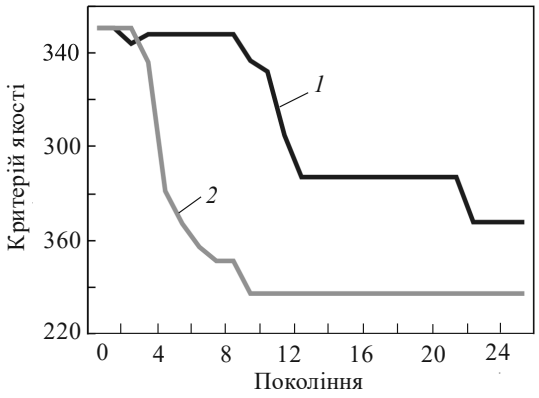

Рис. 2. Графіки залежності отриманої придатності від номера популяції при виконанні першого етапу синтезу: *1* — функція QValue; *2* — функція Roulette

Експериментальні результати. Розроблено програмне застосування SDFCAD для вводу просторового ГСПД та його оптимізації за алгоритмом ГП, що дає змогу задавати такі параметри, як функція придатності, функція відтворення з її ймовірністю мутації, частка елітної множини, розмір популяції, максимальна кількість популяцій.

Оптимізація виконується за два етапи.

На першому етапі генерується перша популяція особин. При цьому ГСПД кожної особини розміщується у тривимірному просторі як множини векторів $\mathbf{K}_i$ та $\mathbf{D}_j$, які мають випадкові координати, але такі, що відповідають умовам (1)—(5). Далі виконується алгоритм ГП з описаними вище функціями Fitness(), Select(), Reproduce(). Досягнувши заданого номера популяції, алгоритм припиняє роботу. В результаті одержуємо популяцію особин, які мають мінімізовану кількість ПЕ (блоків множення, суматорів).

На другому етапі виконується урівноваження матриць *КА*. Для цього у всі дуги просторового ГСПД, крім дуг міжітераційної залежності $\mathbf{D}_{Dj}$, включаються проміжні вершини операторів затримки (регістрів). У результуючій урівноваженій матриці *КА* всі вектори-дуги, крім $\mathbf{D}_{Dj}$, мають координату $t_j = 0$ або $t_j = 1$. Далі за алгоритмом ГП до популяції урівноважених особин застосовуються лише такі операції відтворення як перестановка і рекомбінація. В результаті одержуємо покоління особин з мінімізованою кількістю ПЕ і кількістю регістрів та входів мультиплексорів.

У дослідах з застосуванням SDFCAD використано функцію відбору Select() з назвою Qvalue, яка виконує алгоритм з пропорційним відбором [33]. При цьому *i*-та особина обирається до нового покоління з ймовірністю $P_i = \dfrac{Q_{\max} Q_i}{\Sigma_i (Q_{\max} Q_i)}$, де Q_i — функція придатності *i*-ї особини; $Q_{\max}$ —

Апаратні витрати для обчислення дискретного косинусного перетворення

Процесор	Суматорів	Блоків множення	Регістрів	Входів мультиплексорів
Описаний у роботі [34]	6	3	29	44
Описаний у роботі [35]	7	3	34	16
Запропонований – функція QValue	5	3	31	37
Запропонований – функція Roulette	6	2	32	42

максимальне значення придатності, тобто найгірша придатність. Використовується також функція відбору Roulette, яка виконується за методом Монте-Карло, що вважається різновидом алгоритму з пропорційним відбором.

Як тестовий алгоритм було застосовано алгоритм восьмиточкового дискретного косинусного перетворення, який тестується при застосуванні багатьох методів високорівневого синтезу [34]. Алгоритм містить 13 множень та 29 додавань. За вісім тактів вводяться вісім даних і виводяться вісім результатів через порти вводу і виводу. Покоління налічує 100 осіб, серед яких 10% належать до еліти.

Про роботу алгоритму ГП зазвичай свідчить графік залежності мінімальної функції придатності від номера покоління (рис. 2). Аналізуючи графіки, показані на рис. 2, бачимо, що обидва алгоритми сходяться до деякого локального мінімуму протягом 24 поколінь. Параметри апаратних витрат результуючих конвеєрних процесорів наведено в таблиці. Як бачимо, синтез за функцією Roulette забезпечує мінімальну кількість блоків множення і суматорів внаслідок незначного збільшення кількості конвеєрних регістрів, а відтак і мінімальну вартість обладнання. Слід зауважити, що кількість таких регістрів завжди достатня, коли конвеєрний процесор конфігурується у ПЛІС.

Висновки

Запропонований метод відрізняється від інших методів ГП тим, що за його допомогою можна здійснювати синтез конвеєрних пристроїв для ГСПД зі зворотними зв’язками, виконуючи оптимізацію з урахуванням складності міжз’єднань та мультиплексорів. Результати синтезу конвеєрного пристрою для виконання дискретного косинусного перетворення показали, що запропонований метод забезпечує мінімізовані апаратні витрати, а відтак, мінімізує загальну вартість обладнання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Bhattacharyya S., Wolf M.* Tools and Methodologies for System-Level Design // Electronic Design Automation for IC System Design, Verification, and Testing. L. Scheffer, L. Lavagno, G. Martin — Ed-s. CRC Press. 2016, p. 39—57.
2. *Wang G., Gong W., Kastner R.* Operation Scheduling: Algorithms and Applications // High-level synthesis: from algorithm to digital circuit. P. Coussy, A. Morawiec — Ed-s. Springer. 2008, p. 231—255.
3. *Hwang C.T., Lee T.H., Hsu Y.C.* A formal approach to the scheduling problem in high level synthesis // IEEE Trans. Comput. Aided Des., 1991, Vol. 10, № 4, p. 464—475.
4. *Kruse R., Borgelt C., Braune C. et al.* Computational Intelligence. A Methodological Introduction. 2-nd Ed. Springer, 2016. 564 p.
5. *Miller F.* Cartesian Genetic Programming. Berlin: Springer, 2011. 342 p.
6. *Сергиенко А. М., Симоненко В. П.* Отображение периодических алгоритмов в программируемые логические интегральные схемы // Электрон. моделирование, 2007, **29**, № 2, с. 49—61.
7. 7. *Сергієнко А. М., Сімоненко В. П.* Складання розкладу для графів синхронних потоків даних // Системні дослідження та інформаційні технології, 2016, № 1, с. 51—62. DOI: 10.20535/SRIT.2308-8893.2016.1.06
8. *Affenzeller M., Winkler S., Wagner S., Beham A.* Genetic Algorithms and Genetic Programming. Modern Concepts and Practical Applications. Chapman & Hall, CRC Press, 2009, 358 p.
9. *Ouaiss, I., Govindarajan, S., Srinivasan, V. et al.* An Integrated Partitioning and Synthesis System for Dynamically Reconfigurable Multi-FPGA Architectures. // Proc. of the Reconfigurable Architectures Workshop (RAW'98). Springer, 1998. LNCS, Vol. 1388. Berlin: Heidelberg, p. 31—36.
10. *Grajcar M.* Conditional Scheduling for Embedded Systems using Genetic List Scheduling // Proc. 13th International Symposium on System Synthesis (ISSS). Madrid, Spain, 2000, p. 123—128.
11. *Parsa S., Lotfi S.* A New Genetic Algorithm for Loop Tiling// The Journal of Supercomputing, 2006, Vol. 37, № 3, p.249—269.
12. *Bonsma E., Gerez S.* A genetic approach to the overlapped scheduling of iterative dataflow graphs for target architectures with communication delays // ProRISC Workshop on Circuits, Systems and Signal Processing. November, 27—28, 1997. Mierlo, the Netherlands. 1997, p. 67—76.
13. *Mandal C., Chakrabarti P., Ghose S.* Design space exploration for data path synthesis // Proc. of the 10-th International Conference on VLSI Design, 1996, p. 166—170.
14. *Mandal C., Chakrabarti P., Ghose S.* GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths // IEEE Trans. on VLSI Systems. 2000, Vol. 8, №6, p. 747—750.
15. *Grewal G., O'Cleirigh M., Wineberg M.* An evolutionary approach to behavioural-level synthesis // The 2003 Congress on Evolutionary Computation, December 2003, CEC'03. 2003. Vol. 1, p. 264—272.
16. *Chen D., Cong J.* Register binding and port assignment for multiplexer optimization // Proc. of the 2004 Conf. on Asia South Pacific Design Automation, ASP-DAC'04. 2004, p. 68—73.
17. *Krishnan V., Katkoori S.* A genetic algorithm for the design space exploration of datapaths during high-level synthesis // IEEE Trans. Evolutionary Computation. 2006, Vol. 10, № 3, p. 213—229.

18. *Ferrandi F., Lanzi P.L., Palermo G. et al.* An evolutionary approach to area-time optimization of FPGA designs // Intern. Conf. on Embedded Computer Systems: Architectures, Modeling and Simulation, ICSAMOS. 2007. p. 145—152.
19. *Palesi M., Givargis T.* Multi-objective design space exploration using genetic algorithms // Intern. Symp. on Hardware/software Codesign, CODES. ACM, NY, USA, 2002, p. 67—72.
20. *Pilato C., Tumeo A., Palermo G. et al.* Improving evolutionary exploration to area-time optimization of FPGA designs // Journal of Systems Architecture. 2008, Vol. 54, p. 1046—1057.
21. *Poli R.* Evolution of Graph-like Programs with Parallel Distributed Genetic Programming // Genetic Algorithms: Proc. 7-th International Conference, 1997, p. 346—353.
22. *Miller J. F., Walker J. A.* Embedded cartesian genetic programming and the lawnmower and hierarchical-if-and-only-if problems // Genetic and Evolutionary Computation Conference, GECCO06. July, 2006, Seattle, Washington, USA, p. 911—918.
23. *Parhi K. K.* VLSI Digital Signal Processing Systems. Design and Implementation. Wiley. 1999, 784 p.
24. *Husa J., Kalkreuth R.* A Comparative Study on Crossover in Cartesian Genetic Programming // Proc. 21-st European Conference "Genetic Programming". EuroGP, Parma, Italy, April 4—6, 2018. LNCS. 2018, Vol. 10781, p. 203—219.
25. *Koncal O., Sekanina L.* Cartesian Genetic Programming as an Optimizer of Programs Evolved with Geometric Semantic Genetic Programming // Proc. 22-nd European Conference in Genetic Programming (EuroGP), April 24—26. Leipzig, Germany, 2019, p. 98—113.
26. *Horn J., Nafpliotis N., Goldberg D. E.* A Niched Pareto Genetic Algorithm for Multiobjective Optimization // Proc. of the 1-st IEEE Conf. on Evolutionary Computation. 1994. Vol. 1, p. 82—87.
27. *Holland J. H.* Adaptation in Natural and Artificial Systems. MIT Press, USA, 1992, 228 p.
28. *Petrowski A., Ben-Hamida S.* Evolutionary Algorithms. Wiley & Sons, Inc., Hoboken, New Jersey, USA, 2017.
29. *Lee S., Soak S., Kim K. et al.* .Statistical properties analysis of real world tournament selection in genetic algorithms // Applied Intelligence, 2008. Vol. 28, № 2, p. 195—205.
30. *Sergiyenko A., Serhienko A., Simonenko A.* A method for synchronous dataflow retiming // IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), Kiev, 2017, p. 1015—1018.
31. *Chao L., LaPaugh A., Sha E*. Rotation scheduling: A loop pipelining algorithm // Proc 30-th Design Automation Conf. (DAC'93), June 1993, p. 566—572.
32. *Jin Y.* A comprehensive survey of fitness approximation in evolutionary computation // Soft Computing, 2005, Vol. 9, №1, p. 3—12.
33. *Норенков И. П.* Основы автоматизированного проектирования. М.: Изд-во МГТУ им. Баумана, 2009, 430 с.
34. *Nikara J., Takala J., Akopian D., Saarinen J.* Pipeline Architecture for DCT/IDCT // IEEE International Symposium on Circuits and Systems. ISCAS, 2001, May 6-9, Sydney, Australia, 2001, p. 902—905.
35. *Hsiao S.-F., Shiue W.-R., Tseng J.-M.* A cost efficient fully-pipelinable architecture for DCT/IDCT // IEEE Trans. On Communications, 1991, Vol. 39, № 5, p. 640—643.

Отримано 04.02.20

REFERENCES

1. Bhattacharyya, S. and Wolf, M. (2016), Tools and Methodologies for System-Level Design. Electronic Design Automation for IC System Design, Verification, and Testing, CRC Press.
2. Wang, G., Gong, W. and Kastner, R. (2008), Operation Scheduling: Algorithms and Applications. High-level synthesis: from algorithm to digital circuit, Springer.
3. Hwang, C.T., Lee, T.H. and Hsu, Y.C. (1991), "A formal approach to the scheduling problem in high level synthesis", *IEEE Trans. Comput. Aided Design,* Vol. 10, no. 4, pp. 464-475.
4. Kruse, R., Borgelt, C., Braune, C., Mostaghim, S. and Steinbrecher, M. (2016), Computational Intelligence. A Methodological Introduction. 2-nd Ed, Springer.
5. Miller, F. (2011), Cartesian Genetic Programming, Springer, Berlin, Germany.
6. Sergiyenko, A. M. and Simonenko, V. P. (2007), "Mapping periodic algorithms to programmable logic integrated circuits", *Electronic Modeling,* Vol.29. no. 2, pp. 49-61.
7. Sergiyenko, A.M. and Simonenko, V.P. (2016), "Sheduling of Synchronous Data Flows", *System Investigations and Informational Technologies*, no. 1, pp. 51-62. DOI: 10.20535/SRIT.2308-8893.2016.1.06
8. Affenzeller, M., Winkler, S., Wagner, S. and Beham, A. (2009), Genetic Algorithms and Genetic Programming. Modern Concepts and Practical Applications, Chapman & Hall, CRC Press.
9. Ouaiss, I., Govindarajan, S., Srinivasan, V., Kaul, M. and Vemuri, R. (1998), "An Integrated Partitioning and Synthesis System for Dynamically Reconfigurable Multi-FPGA Architectures", *Proceeding of the Reconfigurable Architectures Workshop (RAW'98)*, *Springer*, *LNCS*, Vol. 1388, Berlin, Heidelberg, pp. 31-36.
10. Grajcar, M. (2000), "Conditional Scheduling for Embedded Systems using Genetic List Scheduling", *Proceeding of the 13th International Symposium on System Synthesis* (ISSS), Madrid, Spain, 2000, pp. 123-128.
11. Parsa, S. and Lotfi, S. (2006), "A New Genetic Algorithm for Loop Tiling", *The Journal of Supercomputing*, Vol. 37, no. 3, pp. 249-269.
12. Bonsma, E. and Gerez, S. (1997), "A genetic approach to the overlapped scheduling of iterative data-flow graphs for target architectures with communication delays", *ProRISC Workshop on Circuits, Systems and Signal Processing*, November 27-28, 1997, Mierlo, Netherlands, pp. 67-76.
13. Mandal, C., Chakrabarti, P. and Ghose, S. (1996), "Design space exploration for data path synthesis", *Proceeding of the 10-th International Conference on VLSI Design*, 1996, pp. 166-170.
14. Mandal, C., Chakrabarti, P. and Ghose, S.(2000), "GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths", *IEEE Trans. on VLSI Systems,* Vol. 8, no. 6, pp. 747-750.
15. Grewal, G., O'Cleirigh, M. and Wineberg, M. (2003), "An evolutionary approach to behavioural-level synthesis", *The 2003 Congress on Evolutionary Computation, (CEC'03),* Vol. 1, pp. 264-272.
16. Chen, D. and Cong, J. (2004), "Register binding and port assignment for multiplexer optimization", *Proceeding of the 2004 Conference on Asia South Pacific Design Automation, (ASP-DAC'04),* pp. 68-73.
17. Krishnan, V. and Katkoori, S. (2006), "A genetic algorithm for the design space exploration of datapaths during high-level synthesis", *IEEE Trans. Evolutionary Computation,* Vol. 10, no. 3, pp. 213-229.

18. Ferrandi, F., Lanzi, P.L., Palermo, G., Pilato, C., Sciuto, D. and Tumeo, A. (2007), "An evolutionary approach to area-time optimization of FPGA designs", *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, (ICSAMOS)*, pp. 145-152.
19. Palesi, M. and Givargis, T. (2002), "Multi-objective design space exploration using genetic algorithms", *International Conference on Hardware/software Codesign, (CODES), ACM*, New York, USA, pp. 67-72.
20. Pilato, C., Tumeo, A., Palermo, G., Ferrandi, F., Lanzi, P. L. and Sciuto, D. (2008), "Improving evolutionary exploration to area-time optimization of FPGA designs", *Journal of Systems Architecture,* Vol. 54, pp. 1046-1057.
21. Poli, R. (1997), "Evolution of Graph-like Programs with Parallel Distributed Genetic Programming", *Genetic Algorithms: Proceeding of the 7-th International Conference*, pp. 346- 353.
22. Miller, J.F., Walker, J.A. (2006), "Embedded cartesian genetic programming and the lawnmower and hierarchical-if-and-only-if problems", *Genetic and Evolutionary Computation Conference, GECCO06*, Seattle, Washington, USA, July, 2006, pp. 911-918.
23. Parhi, K. K. (1999), VLSI Digital Signal Processing Systems. Design and Implementation, Wiley.
24. Husa, J. and Kalkreuth, R. (2018), "A Comparative Study on Crossover in Cartesian Genetic Programming", *Proceeding of the 21-st European Conference "Genetic Programming", EuroGP,* Parma, Italy, April 4-6, 2018, Vol. 10781, pp. 203-219.
25. Koncal, O. and Sekanina, L. (2019), "Cartesian Genetic Programming as an Optimizer of Programs Evolved with Geometric Semantic Genetic Programming", *Proceeding of the 22-nd European Conference in Genetic Programming, EuroGP*, Leipzig, Germany, April 24-26, 2019, pp. 98-113.
26. Horn, J., Nafpliotis, N. and Goldberg, D.E. (1994), "A Niched Pareto Genetic Algorithm for Multiobjective Optimization", *Proceeding of the 1-st IEEE Conf. on Evolutionary Computation*, Vol. 1, pp. 82-87.
27. Holland, J. H. (1992), Adaptation in Natural and Artificial Systems*,* MIT Press, USA.
28. Petrowski, A. and Ben-Hamida, S. (2017), Evolutionary Algorithms, Wiley & Sons, Inc, Hoboken, New Jersey, USA.
29. Lee, S., Soak, S., Kim, K., Park, H. and Jeon, M. (2008), "Statistical properties analysis of real world tournament selection in genetic algorithms", *Applied Intelligence,* Vol. 28, no. 2, pp. 195-205.
30. Sergiyenko, A., Serhienko, A. and Simonenko, A. (2017), "A method for synchronous dataflow retiming", *IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, Kiev, Ukraine, 2017, pp. 1015-1018.
31. Chao, L., LaPaugh, A. and Sha, E. (1993), "Rotation scheduling: A loop pipelining algorithm" *Proceeding of the 30-th Design Automation Conference. (DAC'93)*, June 1993, pp. 566-572.
32. Jin, Y. (2005), "A comprehensive survey of fitness approximation in evolutionary computation", *Soft Computing,* Vol. 9, no. 1, pp. 3-12.
33. Norenkov, I.P. (2009), *Osnovy Avtomatizirovannogo Proektirovaniya* [Computer Aided Design Basics], Izd-vo MGTU im. Baumana, Moskow, Russia.
34. Nikara, J., Takala, J., Akopian, D. and Saarinen, J. (2001), "Pipeline Architecture for DCT/IDCT", *IEEE International Symposium on Circuits and Systems, (ISCAS 2001)*, May 6-9, Sydney, Australia, pp. 902-905.
35. Hsiao, S.-F., Shiue, W.-R. and Tseng, J.-M. (1991), "A cost efficient fully-pipelinable architecture for DCT/IDCT", *IEEE Trans. On Communications*, Vol. 39, no. 5, pp. 640-643.

Received 04.02.20

А.М. Сергиенко, В.А. Романкевич, А.А. Сергиенко

ГЕНЕТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ
СПЕЦИАЛИЗИРОВАННЫХ КОНВЕЙЕРНЫХ УСТРОЙСТВ

Предложен метод синтеза специализированных конвейерных устройств, основанный на генетическом программировании. Метод заключается в представлении алгоритма пространственным графом синхронных потоков данных, кодировке его матрицы вершин-операторов хромосомой и использовании алгоритма генетической оптимизации. Высокая эффективность метода показана на примере синтеза процессора для дискретного косинусного преобразования, который конфигурируется в программируемой логической интегральной схеме.

К л ю ч е в ы е с л о в а: программируемые логические интегральные схемы, граф потоков данных, генетическое программирование.

A.M. Sergiyenko, V.A. Romankevich, A.A. Serhienko

GENETIC PROGRAMMING
OF APPLICATION-SPECIFIC PIPELINED DATAPATHS

A method for the synthesis of application-specific pipeline data paths based on the genetic programming is proposed. The method consists in representing the algorithm with a spatial synchronous data flow graph, encoding its matrix of operator-nodes as chromosomes and using the genetic optimization algorithm. The high efficiency of the method is shown by the example of the discrete cosine transform processor synthesis, which is configured in FPGA.

K e y w o r d s: FPGA, VHDL, SDF, data flow graph, genetic programming.

СЕРГІЄНКО Анатолій Михайлович, д-р техн. наук, ст. наук. співроб., професор кафедри обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського». В 1975 р. закінчив Київський політехнічний ін-т. Область наукових досліджень — архітектура ком'ютерів, високорівневий синтез обчислювальних пристроїв, програмування ПЛІС, цифрова обробка сигналів, штучний інтелект.

РОМАНКЕВИЧ Віталій Олексійович, д-р техн. наук, професор, зав. кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського». В 1996 р. закінчив Київський політехнічний ін-т. Область наукових досліджень — синтез відмовостійких багатопроцесорних систем, штучний інтелект.

СЕРГІЄНКО Анастасія Анатоліївна, асистентка кафедри обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського», який закінчила у 2016 р. Область наукових досліджень — високорівневий синтез обчислювальних пристроїв, цифрова обробка сигналів, штучний інтелект.