УДК 681.518:681.326.7

СКОБЦОВ Ю.О.[1], СКОБЦОВ В.Ю.[2], ІВАНОВ Д.Є[2]

[1]Донецький Державний Технічний Університет, МОН України

[2]Інститут Прикладної Математики та Механіки, НАН України

ЕВОЛЮЦІЙНІ МЕТОДИ ПОБУДОВИ ПЕРЕВІРЯЮЧИХ ТЕСТІВ ДЛЯ ЦИФРОВИХ СХЕМ

В роботі представлено єдиний підхід до генерації тестів для комбінаційних, послідовністних цифрових схем та мікропроцесорних систем, що базується на генетичних алгоритмах та генетичному програмуванні. Розглянуто кодування особнів та проблемно-орієнтовні генетичні оператори кросинговеру та мутації, які використовуються у задачах побудови тестів цифрових схем.

1. Вступ. Суть задачі побудови тестів для цифрових схем (ЦС) полягає в пошуку вхідної послідовності, яка для кожного пошкодження з заданої множини дає різні вихідні значення сигналів у непошкодженій та пошкодженій ЦС. Відомо, що ця задача є *NP-повною*. Детерміновані алгоритми побудови тестів мають високу складність, і у виродженому випадку можуть звестися до повного перебору, для двох головних способів представлення ЦС - структурного у вигляді логічної схеми та функціонального. Особливо це характерно для методів побудови тестів для послідовністних цифрових схем. В останнє десятиліття одержали розвиток нові підходи до рішення задачі побудови тестів – символьний підхід, алгоритми, засновані на моделюванні та еволюційний підхід. Останній забезпечує кращі результати для послідовністних схем великої розмірності в порівнянні з іншими.

В даний час бурхливе розвивається новий напрямок у теорії і практиці штучного інтелекту – еволюційні обчислення – термін, який звичайно використовується для загального опису алгоритмів пошуку, чи оптимізації навчання, заснованих на деяких формалізованих принципах природного еволюційного добору. Еволюційні обчислення використовують різні моделі еволюційного процесу. Серед них можна виділити наступні основні парадигми:

1. Генетичні алгоритми (ГА);
2. Еволюційні стратегії;
3. Еволюційне програмування;
4. Генетичне програмування.

Відрізняються вони, в основному, способом представлення рішень та різним набором використаних у процесі моделювання еволюції операторів. Відзначимо, що в даний час усі парадигми використовуються при генерації тестів цифрових систем.

2.Генетичні алгоритми, будучи однією з парадигм еволюційних обчислень, являють собою алгоритми випадкового спрямованого пошуку для побудови (суб)оптимального рішення даної проблеми, що моделює процес природної еволюції. Класичний “простий” ГА використовує двійкові рядки для кодування рішення проблеми - особи (хромосоми) [1]. Це робить його привабливим для використання в задачах генерації перевіряючих тестів логічних схем,

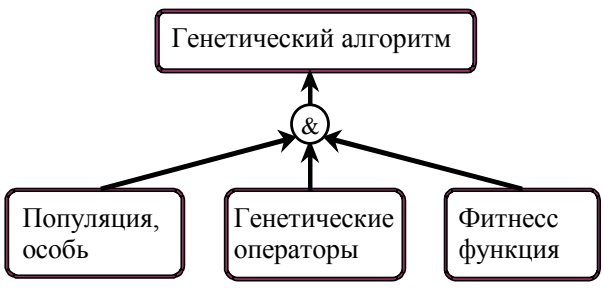


Рис. 1 Основні компоненти ГА.

де рішення задачі представляється у вигляді двійкових наборів чи їхніх послідовностей. На множині рішень визначається цільова (fitness) функція (ЦФ), що дозволяє оцінити близькість кожного особня до оптимального рішення. Простий ГА використовує три основних оператори: репродукція, кросинговер (або схрещення) та мутація. Використовуючи дані оператори, популяція (набір потенційних рішень даної проблеми) еволюціонує від покоління до покоління [1]. Таким чином, щоб задати ГА для рішення конкретної проблеми, необхідно визначити поняття особня, популяції, операцій схрещення і мутації, задати оцінюючу функцію. Очевидно також, що ефективність генетичного алгоритму залежить від цілого ряду параметрів: розміру популяції, стратегії вибору особнів з попередньої популяції, імовірностей схрещування і мутації, стратегії скорочення популяції.

3. Побудова тестів для комбінаційних і послідовністних схем. На першому етапі ГА застосовувалися для генерації тестів комбінаційних ЦС (без пам'яті) [2]. Оскільки значення на вихідних лініях комбінаційної цифрової схем залежать тільки від значень на зовнішніх входах і не залежать від внутрішнього стану схеми, тут у ролі особня виступає одиночний двійковий тестовий набір. Група тестових наборів утворює популяцію. До обраним у такий засіб особням застосовуються стандартні оператори кроссинговера і мутації ГА (рис.2).

Сучасні ЦС мають складну послідовністну структуру. Для зручності моделювання синхронна послідовністна схема перетворюється в псевдокомбінаційний еквівалент шляхом обриву зворотних зв'язків у місцях їхньої синхронізації. Під час генерації тестів таких ЦС із застосуванням ГА в якості особня використовується тестова послідовність (рис.3а), що складається з фіксованого числа тестових наборів, можливо, різної довжини (рис.3б). Для обраного у такий спосіб представлення особнів і популяцій розроблено наступні проблемно орієнтовані генетичні оператори [3,4]:

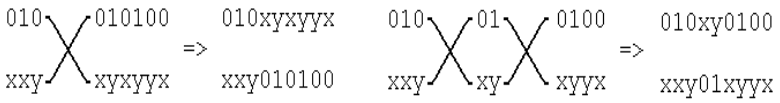
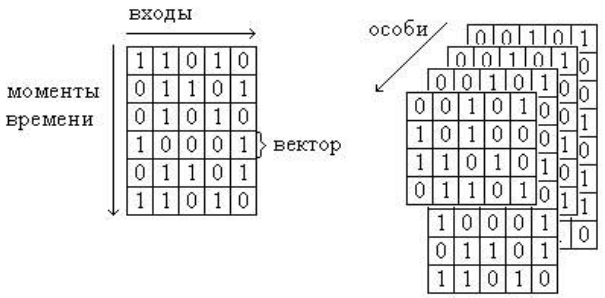


Рис. 2 Одно- та дво-точечный операторы кроссингову.



а) особень б) популяція

Рис.3. Кодування особнів та популяцій в ГА.

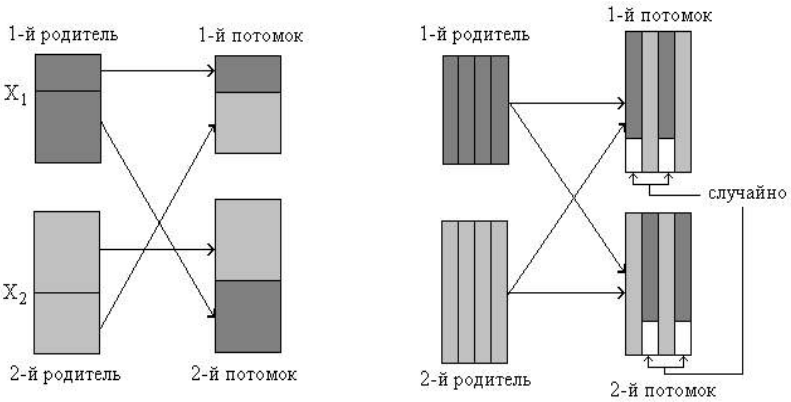


Рис.4 Операції горизонтального та вертикального схрещення у ГА.

1) Схрещування. Реалізується два види операції (рис.4): вертикальне і горизонтальне схрещування, що виконуються відповідно з імовірностями P_v і $P_h = 1 - P_v$.

2) Мутація. Застосовуються три види операції відповідно з імовірностями P_{m_1}, P_{m_2} і P_{m_3}:

– видалення одного вхідного вектора з випадково обраної

позиції. Застосування даної операції дозволяє зменшувати довжину тестової послідовності, що генерується, в тому випадку, коли вилучений вектор не погіршує її тестові властивості;

– додавання одного вхідного вектора у випадкову позицію, що дозволяє розширювати область пошуку рішень;

– випадкова заміна бітів у тестовій послідовності.

Найважливішим компонентом ГА при генерації тестів є форма фітнесс-функції, яка в даному випадку заснована на даних логічного моделювання непошкодженої або пошкодженої схеми. Оскільки метою генерації тестів є побудова послідовності, на якій максимально відрізняються значення сигналів у непошкодженій та пошкодженій схемах, то якість тестової послідовності (fitness-функція) оцінюється як міра відмінності значень сигналів у непошкодженій та пошкодженій схемах. У найпростішому випадку для цього використовуються програми логічного моделювання непошкоджених цифрових схем, що дозволяють оцінити значення сигналів на двох сусідніх (у часі) тестових наборах. На основі отриманих дані моделювання розроблені fitness-функції наступного вигляду [3,4]:

$$h(v,f) = c_1 f_1(v,f) + c_2 * f_2(v,f), \qquad (1)$$

де f_1 - та f_2 - число змін сигналів на виходах логічних вентилів і тригерів відповідно ($c_2 >> c_1$). Цільова функція для послідовності наборів визначається як зважена сума фітнесс-функцій окремих наборів:

$$H(s,f) = \sum_{i=1}^{i=\text{длина}} L^i * h(v_i,f), \qquad (2)$$

де s – послідовність, що аналізується; v_i - вектор з розглянутої послідовності, i – позиція вектора в послідовності, f – задане пошкодження, $0 < L < 1$.

Якщо не вдається побудувати тест із використанням цільової функції зазначеного типу, то застосовуються програми моделювання непошкоджених цифрових схем, що вимагають більших машинних ресурсів. Цільові функції при цьому засновані на підрахунку розходжень сигналів у непошкодженій та пошкодженій схемах і мають приблизно такий же вигляд як і в попередньому випадку. При цьому, $f_1(v,f)$ і $f_2(v,f)$ можуть інтерпретуватися відповідно як зважене число вентилів і тригерів з різними значеннями в непошкодженій та пошкодженій схемах. Нижче наведено укрупнений псевдокод генетичного алгоритму генерації тесту для заданого пошкодження [4].

```
Пошук_перевіряючої_послідовності(пошкодження_ціль)
{
  for( i=0 ; i<МАХ_ПОКОЛІНЬ ; i++)
  {
    for( кожного особня s у популяції P )
      обчислити_оцінку H(s,f);
    new=∅ ;
    for( k=0 ; k<ЧИСЛО_НОВИХ_ОСОБНІВ ; k++ )
    {
      вибрати_двох_особнів_у_P();
      застосувати_операцію_схрещення();//генеруються два особня s
      застосувати_операцію_мутації_до_s_с_імовірністю_Pm();
```

```
        new=new∪ s;
      }
      P=(кращі MAX_ОСОБНІВ з new і P )
      for( кожного особня s у популяції P )
        if( s виявляє f )
          return s;
   }
    return( НЕМАЄ_ПОСЛІДОВНОСТІ )
}
```

4. Генерація тестів для мікропроцесорних систем. При генерації тестів для мікропроцесорних (МП) систем одним із самих перспективних є підхід, заснований на генетичному програмуванні (ГП). Перевіряючою послідовністю для МП системи є тест-програма, що складається з операторів асемблера. Класичне ГП використовує для представлення особня деревоподібні структури, що не дозволяють працювати з довільними програмами. Тому в даному випадку використовується підхід, заснований на представленні програми орієнтованим ациклічним графом (ОАГ) (рис.5) [5]. Кожен вузол такого графа містить покажчик на бібліотеку команд і, якщо це необхідно, на параметри команди (показано на Рис.5 праворуч). Відзначимо, що приведений граф має 4 види вершин:1)пролог; 2)епілог; 3)послідовні (лінійні) вершини; 4)вузли розгалуження. При цьому «пролог» і «епілог» представляють необхідні команди, що виконують, наприклад, ініціалізацію програми. Вони визначаються конкретною МП системою і можуть бути порожні. Послідовні (лінійні) вузли представляють «звичайні» команди (наприклад, арифметичні і логічні – на рис.5 вершини B,F). Відзначимо, що команди безумовного переходу також представляються вершинами цього типу. Вузли розгалуження графа відповідають командам умовного переходу.

Таким чином, тест-програма МП системи генерується шляхом зміни топології ОАГ і мутації параметрів вершини графа. При цьому зазвичай використовується еволюційна ($\lambda + \mu$) – стратегія, що моделює розвиток

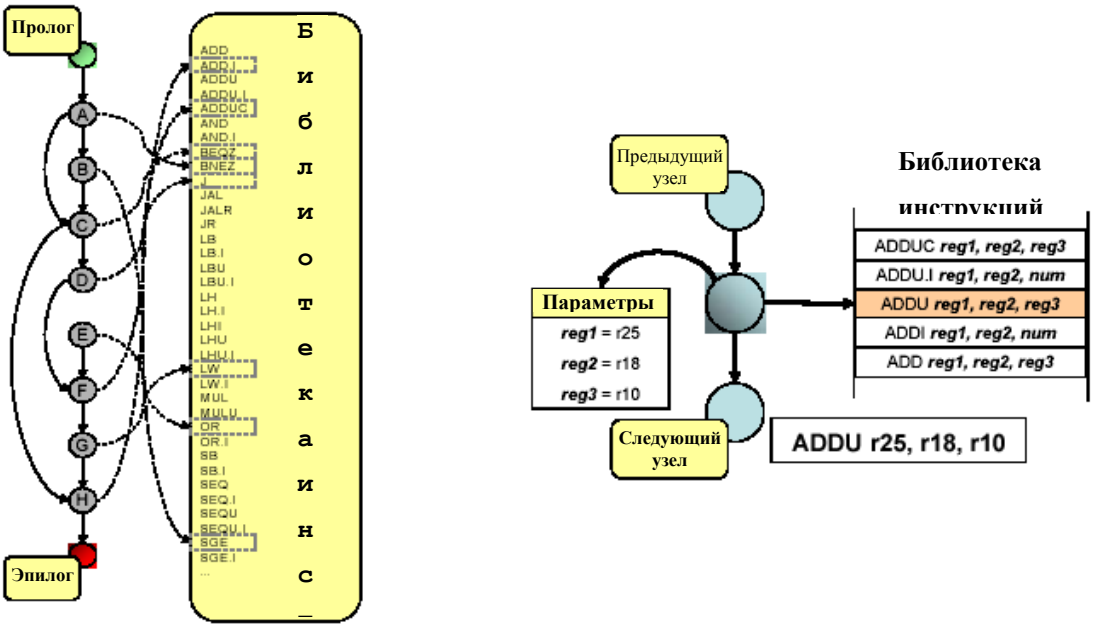


Рис.5 Представлення програми-особня орієнтованим ациклічним графом.

популяція з μ особнів, де кожний особень представляє тест-програму. Вибір батьків виконується методом турнірного добору (тобто вибираються випадково τ особнів і серед них відбирається 1 кращий). Після генерації λ нащадків кращі μ тест-програм відбираються в проміжну популяцію, що містить ($\lambda + \mu$) особнів. У процесі еволюції використовуються наступні генетичні оператори.

Оператори мутації:

- додавання нового оператора у випадкову позицію. Відзначимо, що новий вузол може бути як «лінійним» так і безумовною чи умовною командою переходу. У випадку додавання команди переходу, випадково генерується номер вузла переходу. Відзначимо, що при використанні команди безумовного переходу можуть бути отримані недосяжні вершини;
- видалення вузла у випадковій позиції. Якщо віддаляється вузол переходу (у який іде перехід), то необхідна корекція самого вузла розгалуження.

Оператор кроссинговера – стандартний 1-крапковий ОК. При цьому важливо правильно вибрати крапки перетину, щоб граф розбивався на 2 непересічних підграфа.

При побудові тестів для МП систем, використовуючи описану модель, особні популяції й оператори кроссинговера і мутації, застосовується фітнесс-функція наступного вигляду.

$$F = N_a + N_f * N_m + N_f^2 + N_d \text{ , де}$$

N_f – число всіх пошкоджень,

N_a – число активованих пошкоджень,

N_m – число пошкоджень, що змінюють вміст пам'яті,

N_d – число перевірених пошкоджень.

Таким чином, еволюційний підхід застосовується і до генерації тестів і на рівні МП систем.

5. Висновок. У даній роботі представлено новий підхід до побудови перевіряючих тестів для ЦС, заснований на використанні еволюційних алгоритмів. Показано, що даний підхід може ефективно застосовуватися для побудови тестів на різних рівнях представлення ДУ:

- на структурному рівні – для комбінаційних і послідовністних логічних схем застосовується генетичний алгоритм із проблемно орієнтованими генетичними операторами і фітнесс-функцією;
- на рівні мікропроцесорних систем застосовується підхід на основі генетичного програмування з представленням особня у вигляді ациклічного графу спеціального вигляду та відповідними генетичними операторами.

Література

1. D.E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley Publishing Company, Inc., 1989.
2. Rudnick E.M., Holm J.G., Saab D.G., Patel J.H., Application of Simple Genetic Algorithm to Sequential Circuit Test Generation // Proc. European Design & Test Conf. 1994. P.40-45.
3. Prinetto P., Rebaudengo M., Sonza R.M. An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms // Proc. Int. Test Conf. 1994. P.240-249.

4. Закусило С.А., Иванов Д.Е., Скобцов В.Ю., Скобцов Ю.А. Генетический подход к генерации проверяющих тестов в системе АСМИД-Е // Труды международных конференций "Искусственный интеллект" и "Интеллектуальные САПР". – М.:Физматлит.-2002. – С.49-55.
5. F. Corno, G. Cumani, M. Sonza Reorda, G. Squillero," Fully Automatic Test Program Generation for Microprocessor Cores",DATE2003: Design, Automation and Test in Europe, Munich, Germany, March 3-7, 2003, pp. 1006-1011.