УДК 004.942(045)

АЛГОРИТМ АДАПТАЦІЇ ГРАНИЧНИХ СТРУКТУР

В. М. Опанасенко, д-р техн. наук, проф., ***О. І. Сіянко, А. М. Труш***

Національний авіаційний університет

oleksand.siyanko@mail.ru

Розглянуто принципи побудови граничних структур та наведено їхню математичну модель. Розроблено алгоритм адаптації граничних структур. Виконано програмну реалізацію алгоритму за допомогою мови програмування Java 2 SE 5.0.

Ключові слова: граничні структури, алгоритм адаптації, штучний нейрон, персептрон.

Design philosophy and mathematical model of the boundary structures are considered. Adapting algorithm of the boundary structures is designed. Implemented an application, based on the algorithm, by means of Java 2 SE 5.0.

Keywords: boundary structures, adapting algorithm, artificial neuron, perceptron.

Вступ

Для забезпечення функціонування штучних нейронів (персептронів) [1] необхідно створювати алгоритми, які працюють за універсальними схемами і відповідно потребують менших апаратних ресурсів (до яких належить кількість входів-виходів структури). Одним з таких алгоритмів є алгоритм адаптації граничних структур [2], який виконує розподіл повної множини вихідних двійкових векторів на дві підмножини, залежно від типу граничної функції та значення граничного вектора.

Постановка завдання

Завдання синтезу граничного пристрою полягає у визначенні типів логічних функцій для кожного рівня трикутної матриці (ТМ) (усі логічні елементи (ЛЕ) одного рівня налаштовуються на реалізацію однієї функції), при яких структура реалізує відображення повної множини n-компонентних двійкових векторів.

Алгоритм синтезу граничного пристрою ґрунтується на побітовому аналізі значення граничного вектора відповідно до двійкового представлення даних.

Складовою компонентою персептронів, які реалізують функцію зваженого додавання і порівняння її з граничним значенням, є граничний пристрій.

Залежно від результату порівняння приймається рішення щодо активізації нейрона. Розроблений алгоритм необхідно адаптувати для подальшої апаратної реалізації.

Математична модель

Гранична структура представлена трикутною матрицею з n входами та $(n-1)$ рівнями, кожен з яких є набором універсальних логічних елементів (рис. 1).

Алгоритм адаптації для ТМ ґрунтується на побітовому аналізі значення граничного вектора Ω відповідно до його двійкового представлення $\Omega = \sum_{i=1}^{n} 2^{i-1}\omega_i$ та вибраної граничної функції.

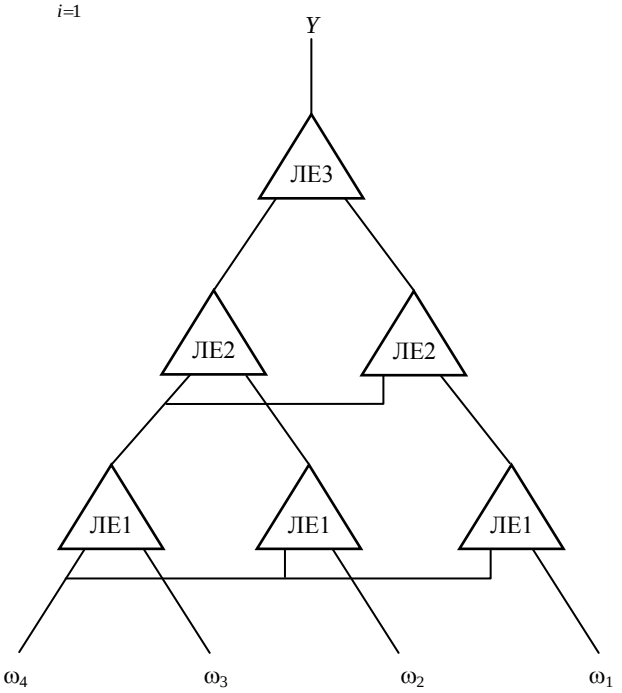

Рис. 1. Узагальнена схема ТМ

Слід зауважити, що аналіз граничного вектора починається зі старших розрядів, при цьому, аналіз двох останніх, молодших розрядів, виконується одночасно.

Залежно від двійкового представлення можна скласти таблиці, за якими буде відбуватися пошук логічних елементів для рівнів ТМ. Позначимо граничні функції таким чином: ψ_1 — операція $\le$, ψ_2 — операція $\ge$, ψ_3 — операція $>$, ψ_4 — операція $<$. Через F_s^j позначимо логічні елементи на рівні s, де $s = 1-(n-2)$ — відповідний рівень ТМ, $j = 1-4$ — індекс граничної функції ψ; а F_{n-1}^j — логічний елемент на рівні $(n-1)$.

Врахувавши вище сказане, складемо таблиці, які використовуються для аналізу граничного вектора (табл. 1, 2).

Таблиця 1

Логічна функція для рівня $s = 1 – (n – 2)$

$\omega_{i=n-s}$	F_s^j
0	$a + b$
1	$a \,\&\, b$

Таблиця 2

Логічна функція для рівня $s = n – 1$

ω_2	ω_1	F_{n-1}^1	F_{n-1}^2	F_{n-1}^3	F_{n-1}^4
0	0	$!(a + b)$	a	$a + b$	$!a$
0	1	$!a$	$a + b$	a	$!(a \,\&\, b)$
1	0	$!(a \,\&b)$	a	$a \,\&\, b$	$!a$
1	1	$!a$	$a \,\&\, b$	a	$!(a + b)$

Однак граничний вектор може набувати особливих значень (1000 або 0111), які потребують спеціальної обробки. У такому разі, для $\forall i \in \{1,2,...,k\}(k > 2)$ при $\omega_i = 0$ та операціях ψ_2 чи ψ_4, або при $\omega_i = 1$ та операціях ψ_1 чи ψ_3, функції F_s^j для $s = 1-(n-k+1)$ визначаються за табл. 1, а для $s = n-k+2$ — за табл. 3.

Таблиця 3

Логічна функція для особливих випадків

$F_s^1 = F_s^3$	a
$F_s^2 = F_s^4$	$\bar{a}$

Опис алгоритму

Алгоритм адаптації граничних структур описує спрощений сценарій порівняння двох операндів. Він дає змогу налаштувати ТМ на виконання граничної функції певного виду з установленим значенням порогу. Поріг — це межа, яка розбиває множину всіх можливих значень, що можуть надходити на входи ТМ, на дві підмножини. Одна з цих підмножин містить значення операндів, які не здатні змінити стан штучного нейрона, а інша — операнди, які збуджують нейрон, тобто переводять його в активний стан.

Для коректної роботи алгоритму, перед його виконанням необхідно вказати такі дані:

- кількість входів ТМ;
- граничний вектор або значення порогу;
- вид граничної операції, на виконання якої має налаштуватися ТМ.

Після того, як було задано початкові дані, згідно з сценарієм алгоритму, здійснюється перевірка порогу щодо особливого випадку, і залежно від того, вибирається гілка алгоритму за якою, порівнево, будуть визначатися типи логічних функцій для відповідних логічних елементів (рис. 2).

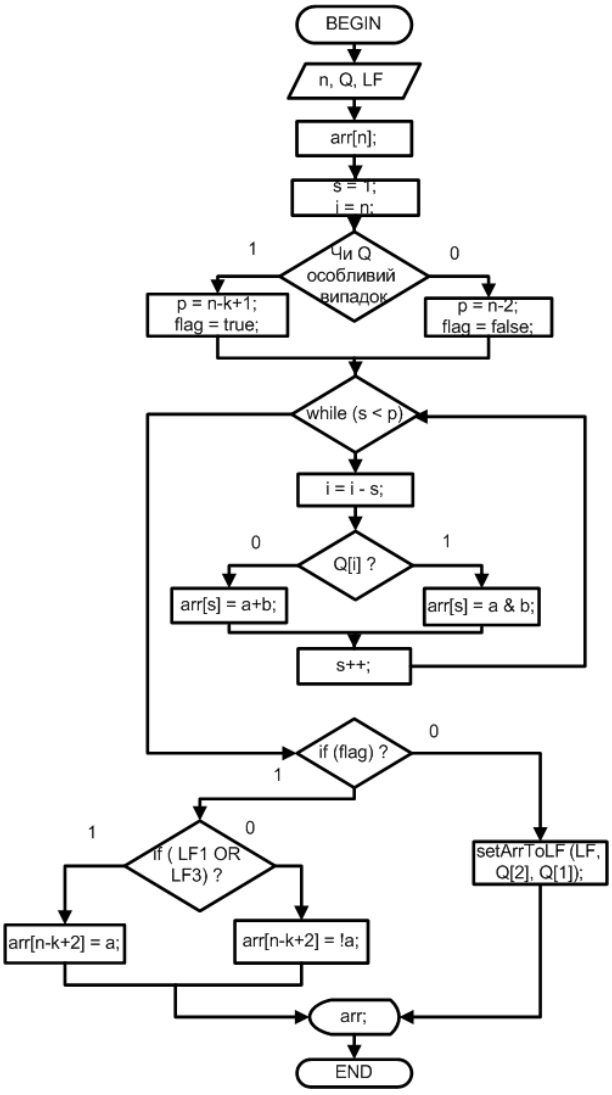

Рис. 2. Алгоритм синтезу елементів граничної структури

Розглянемо роботу алгоритму (рис. 3). Припустимо, що трикутну матрицю, яка має чотири входи, необхідно налаштувати на виконання граничної операції виду $\leq$.

Граничний вектор Ω у двійковій системі числення має значення 0101.

Відповідно до алгоритму адаптації граничних структур аналіз граничного вектора здійснюється зі старших розрядів, тому маємо:

– $\omega_4 = 0$ — тип логічної функції для рівня $s = 1$ $F_s^1 = a + b$;

– $\omega_3 = 1$ — тип логічної функції для рівня $s = 2$ $F_s^1 = a \,\&\, b$;

– $\omega_2 = 0$ та $\omega_1 = 1$ — тип логічної функції для рівня $s = 3$ $F_s^1 = !a$.

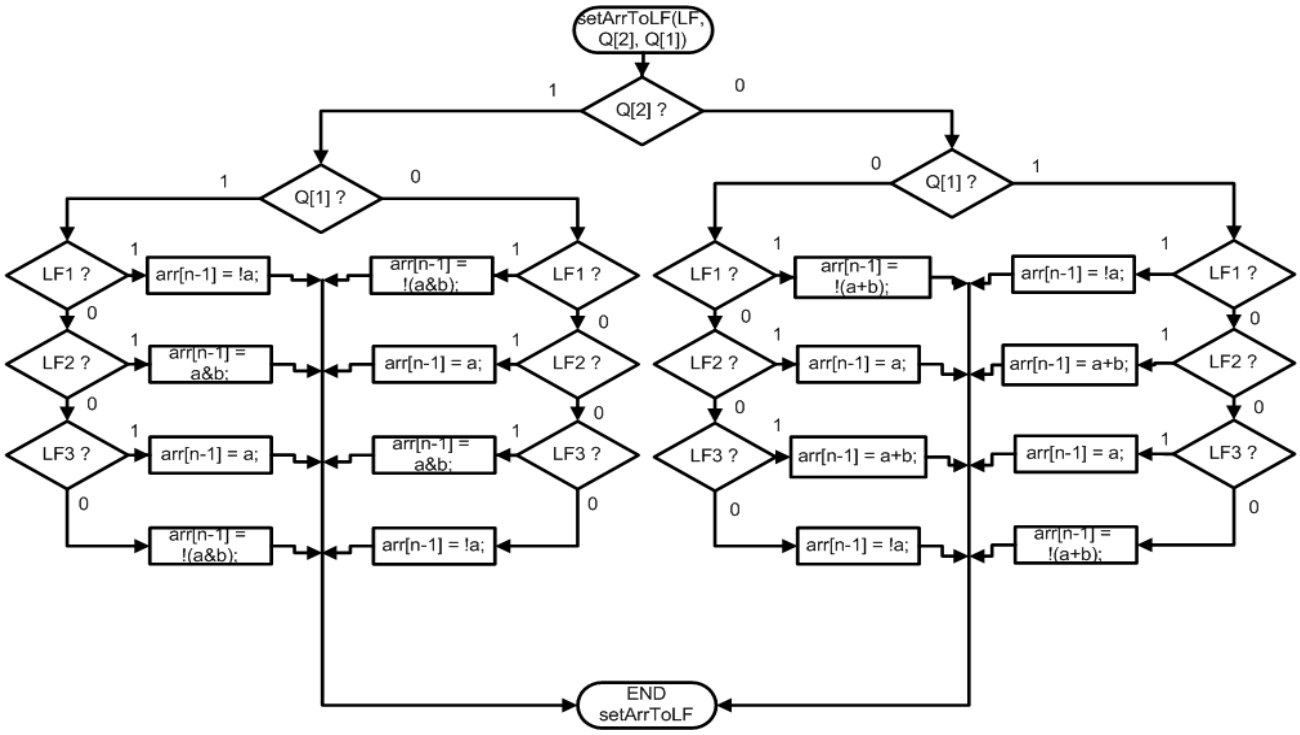

Рис. 3. Процедура *setArrToLF*

Трикутну матрицю, адаптовану до виконання граничної операції виду $\leq$ з порогом Ω, значення якого становить 0101, зображено на рис. 4.

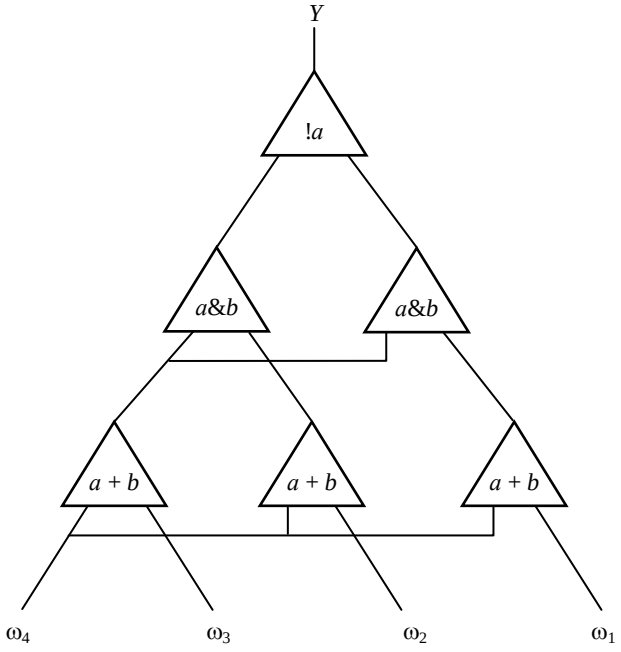

Рис. 4. Структура синтезованої ТМ

Програмна реалізація алгоритму

Оскільки алгоритм граничних структур адаптовано спеціально для апаратної реалізації, доцільно перевірити його роботу, виконавши програмну реалізацію, яка порівняно з апаратною, значно дешевша і дозволяє не менш ефективно, знаходити помилки та виправляти їх.

Програму, яка ґрунтується на цьому алгоритмі, було реалізовано мовою Java 2 SE 5.0 з використанням стандартних пакетів java.lang та java.awt і пакету розширених можливостей графіки javax.swing [3]. Вибір мови програмування був продиктований використанням програми, оскільки передбачалось, що програма буде використовуватися не лише окремо, а й працюватиме через Internet.

Відповідно до об’єктно-орієнтованої моделі програмування програма складається з чотирьох класів, які взаємодіють між собою. Один клас описує логіку роботи алгоритму адаптації граничних структур, а решта три — інтерфейс користувача, який полегшує роботу з програмою.

Програма, реалізована мовою Java, має ряд переваг. По-перше, безпечність даних, яка досягається попередженням виникнення таких ситуацій, як навмисне переповнення стека програми — одного з найпоширеніших способів атаки, який використовується вірусами; пошкодження ділянок пам’яті, які знаходяться за межами простору, виділеного процесу; несанкціоноване читання файлів і їх модифікація. По-друге, незалежність від архітектури комп’ютера. Тобто, програма може виконуватися на будь-яких процесорах; для її роботи необхідна лише віртуальна машина Jav-

. Байт-код програми розроблено таким чином, що він може легко інтерпретуватися, або в процесі роботи трансформуватися в набір команд, орієнтованих на конкретний процесор. По-третє,

висока продуктивність, яка досягається за рахунок байт-коду програми.

Працювати з програмою можна і через мережу Internet, за рахунок використання так званих аплетів — програм для роботи через мережу. Суть роботи такої програми полягає в тому, що користувачі завантажують байт-код програми через мережу і виконують його на власних машинах.

Виконання аплета здійснюється під управлінням web-браузера, який підтримує мову Java і спроможний інтерпретувати байт-код.

До недоліків, пов'язаних з написанням програми мовою Java, можна віднести обов'язкову наявність середовища виконання або віртуальну машину — *Java Runtime Environment* (JRE).

Приклад програмної реалізації алгоритму

У лівій частині вікна програми відображено вихідні параметри граничних структур, які вводяться користувачем, а у правій — результат роботи, розділений на дві частини.

У першій частині представлено логічні елементи для відповідних рівнів ТМ, а в іншій — множину операндів, які можуть надходити на входи ТМ. Припустимо, що трикутну матрицю, яка має чотири входи, потрібно налаштувати на виконання граничної операції виду $\leq$ з порогом, що дорівнює 0101.

Приклад роботи програми проілюстровано на рис. 5.

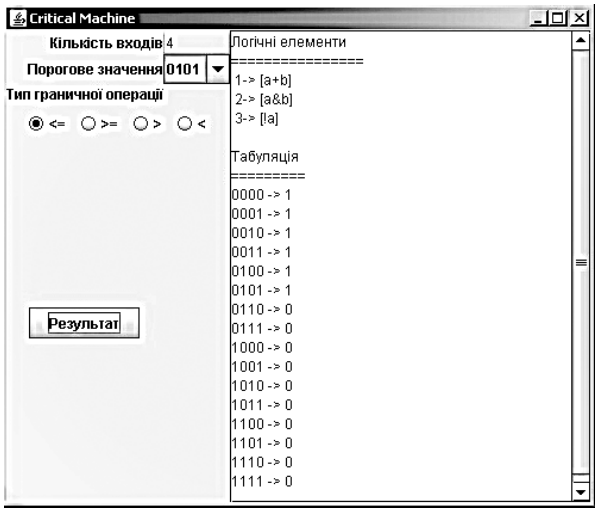

Рис. 5. Вікно програми, реалізованої на базі алгоритму адаптації граничних структур

Повну множину вхідних векторів розбито на дві підмножини. Перша підмножина (значення 0000 – 0101) — це операнди, які збуджують нейрон, інша — операнди (значення 0110 – 1111), які не мають впливу на стан нейрона.

Порівнявши отриманий результат, з тим, який було одержано в прикладі (розділ «Опис алгоритму»), можна зробити висновок, що вони збігаються, а це свідчить про коректність роботи програми.

Висновки

У результаті розробки алгоритму адаптації було синтезовано граничну структуру для реалізації операції порівняння, яка, на відміну від відомих підходів, використовує лише один вхідний операнд.

Програмну реалізацію виконано за допомогою мови програмування Java.

Ця програма дає змогу визначати типи логічних функцій для логічних елементів, на яких будується ТМ, а також відображати результат дії ТМ для повної множини можливих вхідних векторів.

Алгоритм адаптації граничних структур може використовуватися для синтезу граничних структур на основі програмованих логічних інтегральних схем (ПЛІС) типу FPGA.

Його перевагою, порівняно з існуючими алгоритмами, є зменшення кількості контактів кристалу ПЛІС за рахунок відсутності входів для граничного вектора. Програмна реалізація алгоритму мовою Java дає ряд переваг, серед яких: безпека даних, незалежність від архітектури комп'ютера, можливість працювати через мережу Internet та висока продуктивність.

ЛІТЕРАТУРА

1. *Уоссерман Ф.* Нейрокомпьютерная техника: Теория и практика / Ф. Уоссерман. — М. : Мир, 1992. — 240 с.

2. *Палагин А.В.* Реконфигурируемые вычислительные системы / А. В. Палагин, В. Н. Опанасенко. — К. : Просвіта. — 2006. — 295 с.

3. *Хорстманн К. С.* Java2. Библиотека профессионала. — Том 1. Основы, 8-е издание: пер. с англ. / К. С. Хорстманн, Г. Корнелл. — М. : ООО «И. Д. Вильямс», 2008. — 816 с.

Стаття надійшла до редакції 1103.2011.