УДК 004.75

*КРАВЕЦЬ П.І.,*
*ШИМКОВИЧ В.М.,*
*ЗУБЕНКО Г.А.*

# ТЕХНОЛОГІЯ АПАРАТНО-ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ШТУЧНОГО НЕЙРОНА ТА ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ЗАСОБАМИ FPGA

В даній роботі описана методика програмно-апаратної реалізації штучного нейрону з сигмоїдальною функцією активації засобами FPGA, наведено покроковий алгоритм синтезу штучного нейрону та відповідний займаний ресурс FPGA, розроблена технологія побудови штучних нейронних мереж, зокрема таких як RBF- мережі, динамічні мережі Хопфілда та мережі прямого розповсюдження.

In this paper describes methods of software and hardware implementation of artificial neuron with sigmoid activation function by means of FPGA, provides step by step algorithm of artificial neuron and the corresponding resource occupied FPGA, considered technology of artificial neural networks, in particular such as RBF- network, dynamic network Hopfild and feed-forward neural network.

Розвиток теорії автоматичного управління, як і розвиток будь-якого іншого напрямку науки, характеризується ускладненням розв'язуваних завдань і підвищенням якісних показників необхідних рішень [1]. Традиційні методи управління в основному спираються на теорію лінійних систем, в той час як реальні об'єкти є, за своєю природою, нелінійними. Нейромережеві системи управління являють собою новий високотехнологічний напрямок в теорії управління та відносяться до класу нелінійних динамічних систем [2]. Висока швидкодія за рахунок розпаралелювання вхідної інформації в поєднанні зі здатністю до навчання нейронних мереж робить цю технологію вельми привабливою для створення пристроїв управління в автоматичних системах.

На даний час основним методом реалізації нейромережевих систем управління є програмний, з використанням комп'ютерної техніки чи спеціалізованих контролерів, побудованих на її основі, що значно звужує коло практичної реалізації систем управління із-за значної вартості таких регуляторів і робить їх практично недоцільними та недоступними для використання в простих системах керування, крім того комп'ютерні нейромережеві регулятори мають обмежену швидкодію та потребують значних затрат часу на навчання. Рекурентність і послідовність дій процедури навчання нейромережі при її реалізації на всій множині налагоджуваних параметрів не дозволяє повністю вирішити проблему швидкодії процедури навчання нейромережевих структур в темпі з динамікою об'єкта управління. Єдиною альтернативою цьому є розпаралелення процедури навчання та роботи внутрішніх елементів нейромережевих структур. Такі можливості з'являються при апаратно-програмній реалізації нейромережевих структур побудованих на нейрочіпах чи програмованих логічних інтегрованих структурах (ПЛІС-FPGA) [3].

Із сучасних розробок, виконаних на ПЛІС високої інтеграції, можна відзначити, перш за все, «нейрочіп-8», інструментальну плату XDSP-680 на базі FPGA сімейства Spartan компанії Xilinx з нейромережевою прошивкою, що є спільною розробкою Наукового центру нейрокомп'ютерів (Москва) і «Скан Інжиніринг Телеком» (Воронеж). У розробці цих двох партнерів знаходиться і ряд інших виробів: перспективна розробка «нейрочіп-2000» на базі FPGA Virtex / Virtex-E, а також цілий набір інструментальних плат і на базі ПЛІС різних серій і мезонінних модулів різного призначення, що дозволяють швидко і ефективно створювати обчислювальні системи різного функціонального призначення [4, 5].

**Метою даної роботи** є розробка технології та методик апаратно-програмної реалізації нейромережевих структур на FPGA для синтезу нейромережевих регуляторів систем управління, що можуть функціонувати в темпі з процесом управління складними динамічними об'єктами.

Однією з проблем при апаратно-програмній реалізації нейромережевих структур є реаліза-

ція штучного нейрона і його активаційних функцій засобами FPGA.

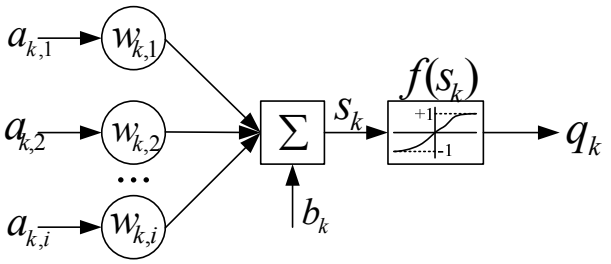


***Рис. 1. Схема штучного нейрону***

де $q_k$ – вихідний сигнал *к*-го нейрона;

$f$ – активаційна функція нейрона;

$a_{k,i}$ – вхідні сигнали *к*–го нейрона;

$w_{k,i}$ – синаптична вага *к*–го нейрона;

$b_k$ – зміщення *к*–го нейрона.

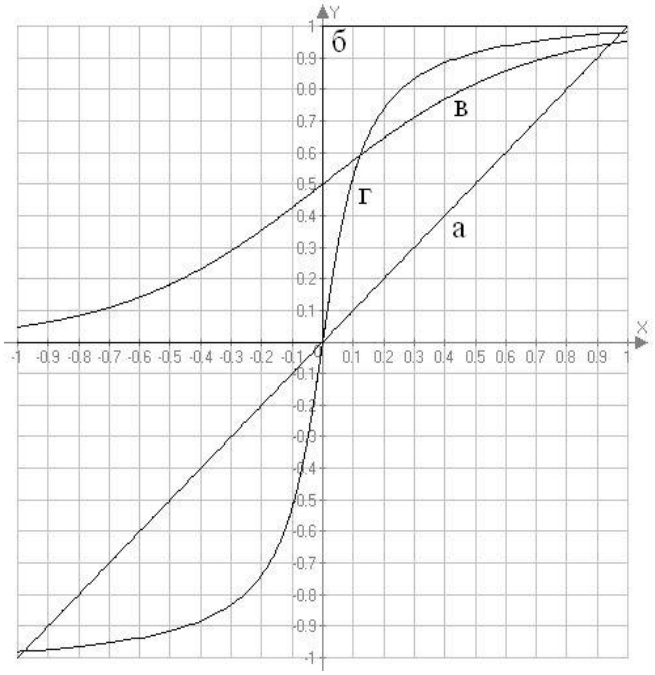


***Рис. 2. Функції активації штучних нейронів***

В штучних нейронних мережах найбільше поширення (по аналогії з біологічними процесами) отримали функції активації такі як лінійна (рис. 2,а), порогова (рис. 2,б) та сигмоїдальна (рис. 2.в).

Сигмоїдальні функція це найбільш ширше використовуваний тип активаційних функцій. Сигмоїдальні функції є монотонно зростаючими, безперервними і диференційованими.

Під сигмоїдальними функціями розуміється клас функцій, які описуються виразом:

$$f(x,k,b,T,c) = k + \frac{c}{1+be^{Tk}}, \qquad (1)$$

де $x,k,b,T,c$ – параметри $k \in R$; $b \in R, b > 0$; $T,c \in R \setminus \{0\}$.

Якщо прийняти *k*=0, *c*=1, *b*=1, і *T*=-1, то вираз (4) прийме вигляд:

$$f(x,0,1,-1,1) = 0 + \frac{1}{1+1e^{-1x}} = \frac{1}{1+e^{-x}}. \qquad (2)$$

Функція, описана виразом (2) називається «класичний» сигмоїд (рис. 2.в).

Якщо прийняти *k*=1, *c*=-2, *b*=1, и *T*=2, то вираз (2) прийме вигляд:

$$f(x,1,1,2,-2) = 1 + \frac{-2}{1+1e^{2x}} = \frac{1+e^{2x}-2}{1+e^{2x}} = \frac{e^{2x}-1}{e^{2x}+1} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (3)$$

Функція, описана виразом (3) називається гіперболічний тангенс (рис. 2.г).

Найскладнішою з погляду цифрової реалізації є сигмоїдальні функції активації.

Існуючі підходи цифрової реалізації нелінійних функцій використовують різні методи апроксимації, такі як табличний метод, розкладання в ряд Тейлора, шматково-лінійна апроксимація і т.д. Розглянемо детальніше сигмоїдальну функцію:

1) $f(x) = \dfrac{1}{1+e^{-x}}$

$$f(-x) = 1 - f(x) = 1 - \frac{1}{1+e^{-x}} = 1 - \frac{1}{1+\frac{1}{e^{x}}} =$$

$$= 1 - \frac{e^{x}}{e^{x}+1} = \frac{e^{x}+1}{e^{x}+1} - \frac{e^{x}}{e^{x}+1} = \frac{1}{e^{x}+1}$$

отримаємо:

$$1 - \frac{1}{1+e^{-x}} = \frac{1}{1+e^{x}}, \quad \text{або} \quad 1 - f(x) = f(-x). \qquad (4)$$

Можемо розглядати $f(x)$ тільки для додатних аргументів. Для від'ємних його значення можна знайти по формулі (4).

2) Для додатних аргументів функція приймає значення в діапазоні від 0,5 до 1. Таким чином якщо розрахувати значення функції з точністю до 2-х знаків після коми можливо всього 50 різних значень (0.50;0.51;0.52;…1.00 ).

Якщо аргумент функції також візьмемо з точністю до 2-х знаків після коми отримаємо таблицю 1:

***Табл. 1. Значення сигмоїдальної функції та її аргументу відповідно***

| $x$ | 0 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | … | 5.29 | 5.30 |
|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 0.50 | 0.50 | 0.50 | 0.51 | 0.51 | 0.51 | … | 0.99 | 1.00 |

Для всіх аргументів $x > 5.3$, $f(x) = 1$. З отриманої таблиці можемо вибрати стовбці в яких значення $f(x)$ змінюється по відношенню до попереднього стовпця та отримати таблицю 2.

***Табл. 2. Оптимізована таблиця значень сигмоїдальної функції та її аргументу відповідно***

| $N$ | 0 | 1 | 2 | 3 | … | 50 |
|---|---|---|---|---|---|---|
| $x$ | 0 | 0.03 | 0.07 | 0.11 | … | 5.30 |
| $f(x)$ | 0.50 | 0.51 | 0.52 | 0.53 | … | 1.00 |

Значення $f(x)$ в отриманій таблиці змінюється на 0.01 від стовпця до стовпця, тому:

1. Таких стовпців буде 50, по кількості можливих значень $f(x)$;
2. Значення $f(x)$ для кожного стовпця можна розрахувати по номеру $N$ цього стовпця за формулою:

$$f(x) = 0.5 + 0.01 \cdot N. \qquad (5)$$

Значення функції $f(x)$ таким чином можливо розрахувати не тільки для одного елемента першої таблиці а для групи елементів, для яких $X_N \le X \le X_{N-1}$, де $X_N, X_{N-1}$ – значення $x$ для $N$, $N$-1 стовпців другої таблиці. Наприклад $x$=0.09, 0.09 знаходиться в між значеннями $0.07 \le 0.09 \le 0.11$, 0.07=$x_2$, 0.11=$x_3$, тобто $N$=2, а $f(x)$=0.5+0.02=0.52.

Таким чином на основі масиву з 50 елементів можемо знайти апроксимацію функції $f(x) = \dfrac{1}{1+e^{-x}}$ для будь-якого аргументу з точністю до 2-х знаків після коми.

Алгоритм реалізації штучного нейрону з сигмоїдальною функцією активації на ПЛІС:

**Крок 1**. Задаємо матрицю синаптичних ваг штучного нейрону $W$, та вектор констант $x$ відповідно до таблиці 2.

**Крок 2**. Визначаємо локальні змінні *lin*, *lout*. Для змінної *lout* задаємо початкове значення 100, що у множині дійсних чисел відповідає одиниці.

**Крок 3**. Змінній *lin* задаємо значення, яке матимемо на виході суматора, що входить до складу штучного нейрона. Це значення можна розрахувати за формулою:

$$lin = \frac{\left(\sum_{i=1}^{n} in_i \cdot W_i\right)}{128}$$

де $n$ – кількість вхідних зв'язків нейрону
$in_i$ – значення на і-тому вході
$W_i$ – вага відповідного входу

**Крок 4**. Для розрахунку функції активації ввести лічильник. Задати $a$=0.

**Крок 5**. Порівняти модуль *lin* з елементами матриці констант ($x$), $x_a$, $x_{a+1}$. Якщо виконується умова $x_a \le |lin| < x_{a+1}$, розрахувати вихідне значення нейрону за формулою (5) *lout*=50+$a$. Перейти на крок 7. Якщо умова не виконується перейти на наступний крок 6.

**Крок 6**. Збільшити a на 1. Якщо $a$=50 перейти на крок 7, інакше – на крок 5. При $a$=50 вхідне значення сигмоїдальної функції більше ніж 5.3, в цьому випадку її вихід дорівнює 1 (початкове значення).

**Крок 7**. Якщо *lin*<0, змінити вихідне значення відповідно формулі (4): *lout*=100-*lin*.

**Крок 8**. Задати значення вихідного сигналу штучного нейрона рівним отриманому значенню локальної змінної.

Побудований штучний нейрон з сигмоїдальною функцією активації на ПЛІС за даною технологією з кроком квантування 0.01 зайняв 765 LUTs (Look Up Table – вентилі логічної матриці). Похибка абсолютна ±0.005. Також було промодельовано штучні нейрони з сигмоїдальною функцією активації на ПЛІС за даною технологією з кроком квантування 0.05 та 0.1 з такими крокоми було використано ресурсу ПЛІС 275 та 179 LUTs відповідно. При зменшенні чи збільшенні кроку квантування значень таблиці 2 буде змінено відповідно і використаний ресурс ПЛІС для одного нейрону.

Далі розглянемо технологію побудови штучних нейронних мереж на ПЛІС.

Однією з головних особливостей нейромереж є паралельна обробка сигналів. Багатошарові нейронні мережі представляють собою однорідну обчислювальну середу. По термінології нейроінформатики це універсальні паралельні обчислювальні структури, призначенні для вирішення самих різних класів задач. Розглянемо

концепцію реалізації нейронних мереж на ПЛІС.

На сьогоднішній день розроблено та досліджено декілька десятків типів штучних нейронних мереж, але основними, принципово різними типами є три типи мереж, що відповідають трьом методам їх навчання, це RBF-мережі, динамічні мережі Хопфілда та мережі прямого розповсюдження.

Основна структура багатошарових нейронних мереж прямого розповсюдження зображена на рис. 3. У повнозв'язних нейронних мережах прямого розповсюдження виходи базових елементів кожного шару сполучені зі всіма входами всіх базових елементів наступного шару.

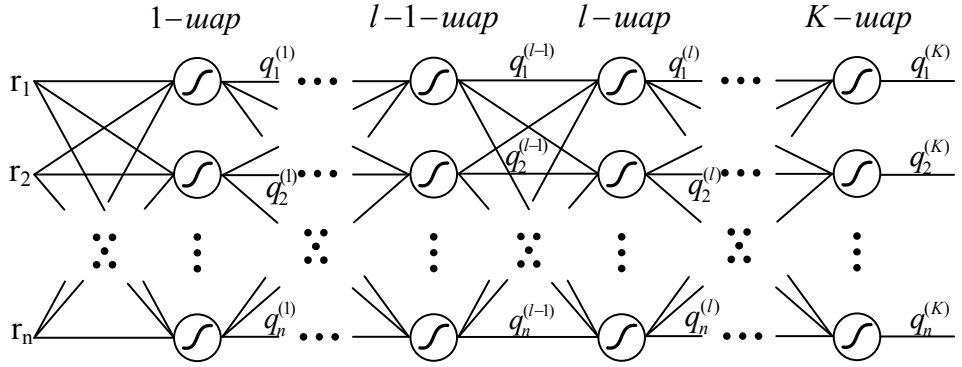


***Рис. 3. Структурна схема багатошарової нейронної мережі прямого розповсюдження***

У силу теорем Колмогорова-Арнольда та Хехт-Нільсена [7] трьох шарів, вхідний, вихідний та один прихований шар, достатньо для одержання ефективного рішення за допомогою нейронних мереж прямого поширення.

При апаратно-програмній реалізації багатошарових нейронних мереж прямого розповсюдження для всіх нейронів з однаковою нелінійною функцією активації використовується одна і та ж матриця констант, для апроксимації цієї функції. Кожний нейрон представляється на мові VHDL окремим процесом. Мова програмування дозволяє явно вказувати сигнали, які запускають процес. Для запуску нейрона використовується вхідний сигнал цього нейрона. Було промодельовано такі нейромережі, і відповідно займаний ними ресурс FPGA в кількості вентилів логічної матриці LUTs: 1-1-1 – 797, 1-2-1 – 829, 2-2-1 – 1728, 3-2-1 – 2544, 4-2-1 – 3497, 1-3-1 – 941, 1-4-1 – 959, 1-2-2 – 940, 1-2-3 – 970, 1-2-5 – 996. Нейронні мережі представлені трьома числами, де перше кількість нейронів в вхідному шарі, друге кількість нейронів в прихованому шарі і третє кількість нейронів в вихідному шарі. Отримані значення в LUTs можуть дещо змінюватись при зміні констант, які задають синаптичні ваги нейронів.

Мережі зі зворотним розповсюдженням відрізняються від нейромереж прямого розповсюдження наявністю каналів, по яких інформація надходить і у зворотному напрямі від виходів на входи буферного шару. Прийнято виділяти два класи багатошарових нейромереж зі зворотним розповсюдженням: рекурентні та рециркуляційні.

Рекурентні нейромережі також передають сигнали у зворотному напрямі, але по каналах зворотного зв'язку. На даний час розроблено і досліджено ряд модифікацій рекурентних мереж. Найбільше поширення серед рекурентних мереж набули релаксаційні мережі із зворотними зв'язками, названі мережами Хопфілда [2].

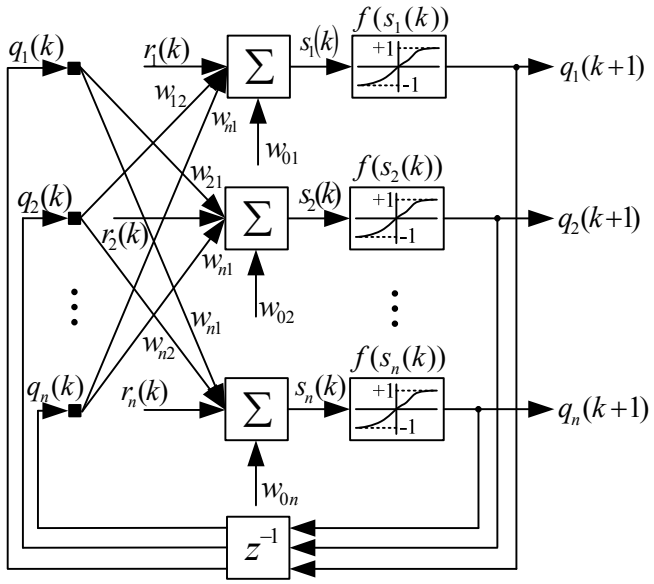

***Рис. 4. Структурна схема одношарової мережі Хопфілда***

У мережах Хопфілда (рис. 4) відбувається динамічне перетворення в часі вхідного сигналу $\mathbf{r} \in R^n$ в $\mathbf{q} \in R^{N_K}$. Структурна схема динамічної мережі Хопфілда зображена на рисунку 4.

У векторно-матричній формі статика мережі Хопфілда описується співвідношенням:

$$\mathbf{s}(k) = \mathbf{W}\mathbf{q}(k) + \mathbf{r} - \mathbf{w}_0,$$

де компоненти вектора функції s($k$) обчислюються як:

$$s_i(k) = \sum_{j=1, j\neq i}^{n} w_{i,j} q_j + r_i - w_{0i}, \quad i = \overline{1,n}.$$

Динаміка мережі Хопфілда забезпечується введенням зворотних зв'язків з виходів $q_j(k)$ на входи базових елементів ($i \neq j$) через елементи затримки $z^{-1} = e^{-\Delta ts}$ ($\Delta t$ – період дискретизації безперервних функцій $q_j(k)$).

Апаратно-програмна реалізація нейронних мереж Хопфілда на FPGA відрізняється від апаратно-програмної реалізації багатошарових нейронних мереж прямого розповсюдження введенням додаткових зворотних зв'язків і елементів затримки в часі на цих зв'язках, що в свою чергу займає більше простору на FPGA. Було змодельовано нейромережі Хопфілда з одним шаром і трьома нейронами в ньому і двома шарами по три нейрони в кожному використаний ресурс FPGA становив відповідно 2.521 і 4.945 LUTs.

RBF-мережі є також універсальними апроксиматорами і при незначних обмеженнях можуть бути застосовані для апроксимації будь-якої безперервної функції [2, 8, 9]. RBF-мережі по своїй структурі відносяться до двошарових мереж, в яких використовується прихований шар з фіксованим нелінійним перетворенням вектора входу з постійними ваговими коефіцієнтами. Цей шар здійснює статичне відображення вхідних змінних $\mathbf{r} \in R^n$ у нові змінні $\mathbf{q}_1^{(l)} = col(q_1^{(l)}, ..., q_m^{(l)})$. Другий, лінійний вихідний, шар «зважує» ці змінні з вагами, що налагоджуються, $w_i = col(w_{i,1}, w_{i,2}, ..., w_{i,m})$, таким чином, що, наприклад, $i$-й скалярний вихід RBF-мережі записується у вигляді:

$$q_i^{(2)} = q_i = F(\mathbf{r}) = \sum_{j=1}^{m} w_{i,j} f_i(\mathbf{r}) + w_0 \cdot 1, \quad i = 1,2,...,K$$

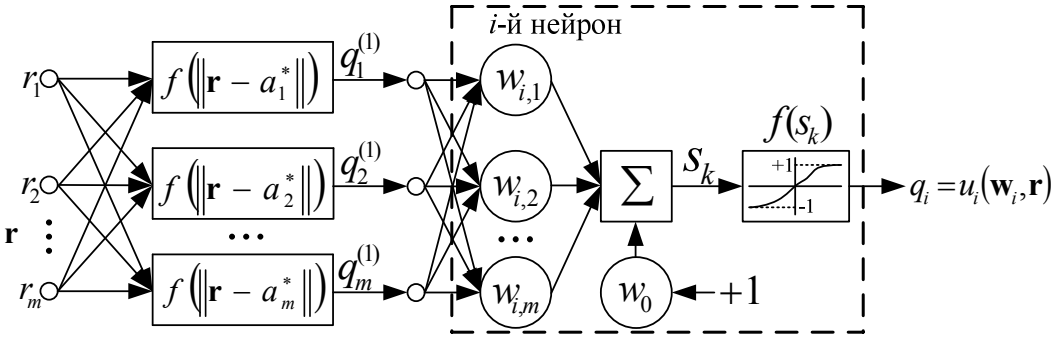


***Рис. 5. Структура RBF-мережі***

Структура RBF-мережі з вектором входу $\mathbf{r} \in R^n$ і вектором виходу $\mathbf{q}^{(2)} = \mathbf{q} \in R^n$ зображена на рисунку 5.

В якості функцій $f(z)$ на практиці найбільше використовується функція Гауса:

$$f_j(\mathbf{r}) = \exp\left( \sum_{k=1}^{n} 0.5\sigma_{j,k}^{-2} \left\| \mathbf{r} - \mathbf{a}_j^* \right\|^2 \right),$$

де $a_j^* = col(a^*_{j,1}, a^*_{j,2}, \cdots, a^*_{j,n})$ – числовий вектор, а $\sigma_{j,k}$ – « ширина» функції Гауса. Специфіка RBF-мережі полягає у тому, що в робочому режимі в них налагоджуються тільки вагові коефіцієнти лінійного вихідного шару. Помилка апроксимації обчислюється безпосередньо на виході мережі так само, як і в БНМ, але налагодження тільки одного, лінійного по параметрах налагодження шару знімає проблему пошуку глобального мінімуму функціонала навчання (як правило, квадратичного) і сприяє швидкій збіжності процесу навчання мережі.

Проблемою RBF-мереж є вибір числа радіально-базисних функцій, необхідних для апроксимації, і ця обставина стає критичним чинником у використовуванні RBF-мереж в задачах ідентифікації і управління, де необхідна інформація для визначення розмірності RBF-мереж як правило, відсутня. Число необхідних радіально-базисних функцій росте експоненціально із зростанням числа вхідних змінних, тобто із зростанням $n$. Звідси витікає висновок про застосовність RBF-мереж в тих задачах, де розмірність вхідної множини (вектора $\mathbf{r}$) обмежена і відома (в БНМ таке обмеження відсутнє). Інша проблема застосування RBF-мереж полягає в необхідності попереднього налагодження прихованого шару. Також однією з основних про-

блем при програмно-апаратній реалізації таких нейромережевих структур є реалізація прихованого шару та нелінійних функцій активації $f_i(\mathbf{r})$, зокрема реалізації функції Гауса.

Для реалізації функції Гауса, виведемо її через сигмоїду. Така реалізація функції Гауса дозволить використовувати одну і ту ж таблицю констант для функції активації нейрона вхідного шару RBF-мережі та функції активації нейрона вихідного шару RBF-мережі, для більш оптимального використання ресурсу FPGA.

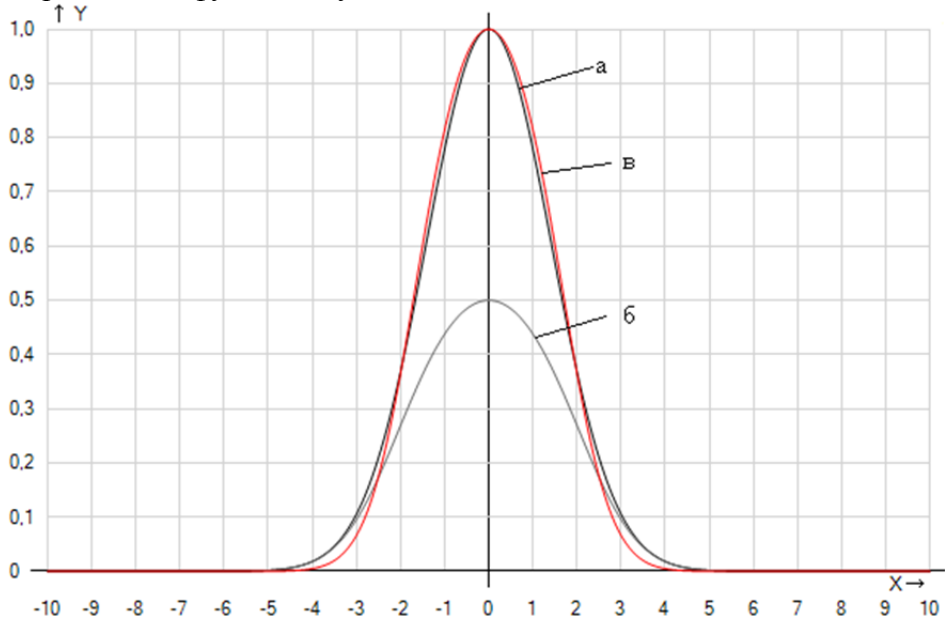


***Рис. 6. Функція Гауса***

Графік функції Гауса зображено на рисунку 6.а, а описується вона так:

$$y = \exp\left(-\frac{x^2}{\sigma^2}\right),$$

Якщо записати замість експоненти сигмоїду, отримаємо графік зображений на рисунку 6.б:

$$y = \mathrm{sigm}\left(-\frac{x^2}{\sigma^2}\right),$$

Якщо модернізувати цю формулу за допомогою коефіцієнтів при $x$ і $\sigma$ та помножити її на 2, отримаємо функцію схожу на функцію Гауса (рис. 6.в):

$$y = 2 \cdot \mathrm{sigm}\left(-\frac{192 \cdot x^2}{128 \cdot \sigma^2}\right)$$

Максимальне відхилення від значення функції Гауса, при даній інтерпретації ≈0.0404.

В якості $x$ в дану функцію передається значення $x = \|\mathbf{r} - \mathbf{a}\|$, де $\|\mathbf{r} - \mathbf{a}\|$ – норма вектора. Норму вектора можливо розраховувати різними методами, але найчастіше використовується Евклідова норма:

$$\mathbf{x} = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\sum_{i=1}^{n} (x_i - a_i)^2},$$

тоді

$$x^2 = \sum_{i=1}^{n} (x_i - a_i)^2$$

Таким чином функція активації нейрона вхідного шару RBF-мережі:

$$f(r) = 2 \cdot \mathrm{sigm}\left(-\frac{192}{128} \cdot \frac{\sum_{i=1}^{n} (r_i - a_i)^2}{\sigma^2}\right). \qquad (6)$$

Згідно (6) необхідно виконати операцію ділення на $\sigma^2$. На мові VHDL операція ділення добре працює тільки коли дільник рівний степені 2. Для того щоб реалізувати операцію ділення на довільне число $\sigma^2$ $\left(y = \frac{x}{\sigma^2}\right)$ можна виконати операцію:

$$\frac{x}{\sigma^2} = \frac{x \cdot sigma}{2^{13}} = \frac{x \cdot sigma}{8192}$$

*sigma* можна розрахувати за формулою:

$$sigma = \frac{8192}{\sigma^2},$$

тоді

$$\sigma^2 = \frac{8192}{sigma} \qquad (7)$$

Якщо задати константу $\sigma^2$ в такому вигляді можна використовувати стандартні операції

мови VHDL. Якщо підставити (7) в (6) отримаємо :

$$f(r)=2\cdot\mathrm{sigm}\left(-\frac{192}{128}\cdot\frac{\sum_{i=1}^{n}(r_i-a_i)^2\cdot sigma}{8192}\right)$$

Таким чином вхідне значення нейрона:

$$lin=\frac{192}{128}\cdot\frac{\sum_{i=1}^{n}(r_i-a_i)^2\cdot sigma}{8192} \quad (8)$$

а функція активації:

$$lout=2\cdot sigm(-lin) \quad (9)$$

З (8) видно що *lin* завжди додатне. Тому сигмо їда у (9) завжди обчислюється для від'ємного числа.

Формула (6) дозволяє обчислити значення для додатного числа. Врахувавши (4) отримаємо аналогічну формулу для від'ємних чисел:

$$f(-x)=1-f(x)=1-(0.5+0.01\cdot N)=0.5-0.01\cdot N$$

Отже згідно (9):

$$lout=2\cdot f(-x)=1-0.02\cdot N \quad (10)$$

Далі розглянемо алгоритм реалізації нейрону прихованого шару з функцією Гауса на ПЛІС.

**Крок 1**. Задаємо вектор констант *x* відповідно до таблиці 2, *a* і константу *sigma*.

**Крок 2**. Визначаємо локальні змінні *lin*, *lout*. Для змінної *lout* задаємо початкове значення 0.

**Крок 3**. Змінній *lin* задаємо значення згідно (8).

**Крок 4**. Для розрахунку функції активації ввести лічильник. Задати *b*=0.

**Крок 5**. Порівняти *lin* з елементами вектору констант *x*. Якщо виконується умова $x_b \le lin < x_{b+1}$, розрахувати вихідне значення нейрону за формулою (10) *lout*=100 – 2*b*, перейти на крок 7, інакше перейти на наступний крок.

**Крок 6**. Збільшити *b* на 1. Якщо *b* =50 перейти на крок 7, інакше – на крок 5. При *b* =50 вхідне значення функції меньше ніж -5.3, в цьому випадку її вихід дорівнює 0 (початкове значення).

**Крок 7**. Задати значення вихідного сигналу штучного нейрона рівним отриманому значенню локальної змінної.

На FPGA промодельовано RBF-мережу з трьома вхідними нейронами з функцією активації Гауса і одним нейроном з сигмоїдальною функцією в вихідному шарі. Така нейромережа зайняла 3.137 LUTs.

## Висновок

В даній роботі описана методика програмно-апаратної реалізації штучного нейрону з сигмоїдальною функцією активації засобами FPGA, наведено покроковий алгоритм синтезу штучного нейрону та відповідний займаний ресурс FPGA. Побудований штучний нейрон з сигмоїдальною функцією активації на ПЛІС за даною технологією з кроком квантування 0.01 зайняв 765 LUTs. Похибка абсолютна ±0.005. Розроблена технологія побудови штучних нейронних мереж, зокрема таких як RBF-мережі, динамічні мережі Хопфілда та мережі прямого розповсюдження. Розробка та моделювання роботи штучного – нейрону та нейроних мереж проходило на програмному забезпеченні Xilinx ISE Design Suite 13.2 та чіпі сімейства Spartan 3 – XC3S200.

## Список літератури


1. Егупов Н.Д. Методы робастного, нейро-нечеткого и адаптивного управления. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.
2. Терехов В.А. Нейросетевые системы Управления: Учеб.пособие для вузов. – М: Высшая школа. 2002. -183с.
3. Соловьев В. Проектирование цифровых систем на основе ПЛИС. – М.: Радио и связь, 2003. – 376с.
4. Гильгурт С.Я. Анализ применения реконфигурируемых вычислителей на базе ПЛИС для реализации нейронных сетей // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Вип. 37. – Київ: 2006. – С. 168-174.
5. Логовский А. Технология ПЛИС и ее применение для создания нейрочипов. // Открытые системы. – 2000. – № 4. – С. 100-102.
6. Barren A. R. Universal approximation bounds for superposition of a sigmoidal function // IEEE Trans. Inform. Theory. 1993. Vol. 39. P. 930 – 945.
7. Саймон Хайкин. Нейронные сети. Полный курс. – М.: Вильямс, 2006. – 1104с.
8. Сигеру Омату. Нейроуправление и его приложения. – М.: ИПРЖР, 2000. – 272с.
9. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. – М.: Горячая линия – Телеком, 2007. – 452 с.