РАДІОЕЛЕКТРОНІКА ТА РАДІОЕЛЕКТРОННЕ АПАРАТОБУДУВАННЯ

https://doi.org/10.31649/1997-9266-2025-181-4-186-194

УДК 004.4:004.04:004.09

С. В. Яковин[1]
С. І. Мельничук[1]
І. З. Мануляк[1]

РЕАЛІЗАЦІЯ ДВОВХОДОВОГО ДИСКРЕТНОГО ПЕРЦЕПТРОНА ЗІ ЗМІЩЕНИМИ СИНАПТИЧНИМИ СИГНАЛАМИ НА ПЛІС ЗАСОБАМИ ALTERAHDL

[1]Івано-Франківський національний технічний університет нафти і газу

Запропоновано й експериментально досліджено апаратну реалізацію дискретного двовходового ймовірнісного перцептрона на програмованій логічній інтегральній схемі (ПЛІС). Перцептрон побудовано з трьох елементарних модулів — shift2b (зміщення синаптичних сигналів простим додаванням), cnts (агрегація на основі підрахунку кількості унікальних значень) і cmp2b (активаційний двобітовий компаратор). Апаратна реалізація дискретного перцептрона ґрунтується на використанні зміщення вхідних сигналів шляхом використання цілочислової операції додавання, що дозволяє знизити апаратні потреби.

Запропонована архітектура реалізації перцептрона забезпечує мінімальну компонентну складність (2—3 логічні елементи на блок) та дозволяє, шляхом лише зміни ваг і порога, відтворити шість базових булевих операцій — OR, AND, XOR, NOR, NAND, XNOR. Такий підхід в перспективі дозволяє створювати апаратно моноструктурні компоненти на основі єдиного блока, що в залежності від потреб може реалізовувати різні логічні функції.

Функціональне моделювання підтвердило коректність реалізації всіх заданих таблиць істинності, а інструментальний аналіз затримок показав критичний шлях 16,7 нс, що відповідає робочій частоті близько 60 МГц без конвеєризації. Отримані аналітичні співвідношення демонструють можливість зменшення апаратних ресурсів порівняно з традиційним лінійним суматором з синтезом логічних функцій першого та другого порядку.

Запропонований підхід відкриває можливості масштабування на більшу кількість входів, інтеграції статистичних (ймовірнісних) критеріїв агрегації та розробки вбудованих процедур он-чіп-навчання. Результати підтверджують перспективність використання дискретних перцептронних структур як легковагових, енергоефективних класифікаторів у системах реального часу та спеціалізованих нейромережевих компонентах.

Ключові слова: перцептрон, бінарні сигнали, булеві функції, опрацювання сигналів, програмовані логічні інтегральні схеми (ПЛІС), нейронні мережі.

Вступ

Традиційно апаратна реалізація штучних нейронних мереж, що ґрунтуються на перцептронних структурах, передбачає використання спеціалізованих цифрових компонентів, оптимізованих для виконання багатопотокових обчислень, зокрема графічні процесори (GPU), тензорні процесори (TPU) тощо. До того ж розробляються нейроморфні (Neuromorphic) процесори (N-M), які імітують роботу відповідних біологічних структур, що сприяє підвищенню продуктивності, а також зменшенню енергоспоживання. Особливістю апаратної реалізації мережевих структур на основі перцептронів є можливість паралельного виконання операцій, що дозволяє збільшити швидкість роботи такої мережі, а також її навчання. Сучасні технології, зокрема програмовані логічні інтегральні схеми (ПЛІС, або FPGA) та спеціалізовані інтегральні схеми (ASIC), дають змогу створювати високопродуктивні апаратні рішення завдяки можливості реконфігурації внутрішніх компонентів.

Це зокрема, дозволяє змінювати архітектуру мережі без необхідності повторного проєктування друкованої плати (PCB). Основні платформи, на яких імплементують штучні нейронні мережі з типовими характеристиками подано в табл.1[1]—[6].

Таким чином, вдосконалення наявних та розробка нових апаратних рішень компонентів нейронних мереж є одним з критично важливих напрямків розвитку сучасних технологій штучного інтелекту.

Таблиця 1

Критерій	MCU	CPU	GPU	FPGA	ASIC	N-M
Продуктивність	KOps	GOPS-TOPS	T/P FLOPS	залежний конвеєр	T/P FLOPS	асинхронна
Затримка	~1…10 мс	>1 мс	10-100 мкс	<1 мкс	10…100 мкс	<100 нс
Енергоспоживання, Вт	10^{-3}	10^{1}	10^{2}	10^{1}	10^{2}	10^{-3}
Гнучкість	висока	висока	середня	максимум	фіксована	специфічна

Інтеграція перцептронних структур, зокрема дискретних, у спеціалізовані мікросхеми створює нові можливості для розробки компактних і ефективних систем штучного інтелекту а також дозволяє розширити їхній функціонал.

Запропоноване, а також досліджене в роботах [7]—[9], структурне рішення дискретного перцептрона зі зміщеними синаптичними сигналами представляється трьома умовно-окремим компонентами (рис. 1):

– компонент перетворення вхідних сигналів (Input Transformation);

– компонент агрегування (Aggregation);

– компонент формування результату (Output Transformation чи Activation).

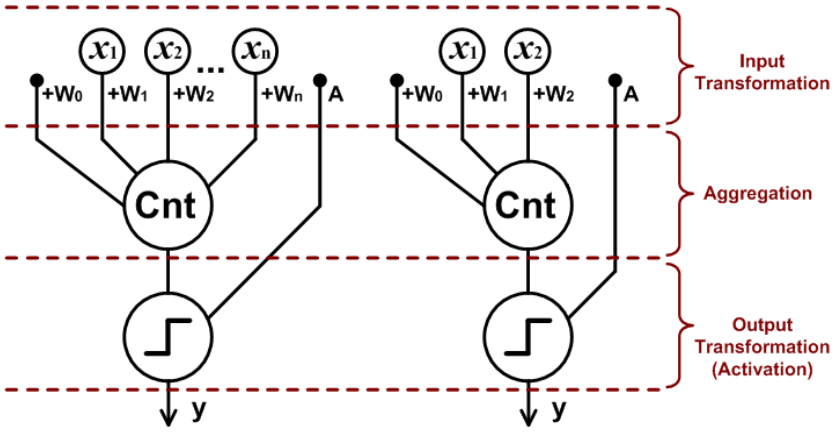


Рис. 1. Узагальнена та адаптована (на два синаптичні вхідні сигнали) структура функціональних компонентів дискретного перцептрона

Input Transformation — реалізує перетворення вхідних сигналів, зокрема для дискретного перцептрона виконується зміщення вхідних синаптичних сигналів на основі операції додавання, що суттєво знижує апаратні потреби на реалізацію такого компоненту.

Aggregation — реалізує опрацювання перетворених вхідних сигналів, зокрема для дискретного перцептрона виконується оцінювання інформаційної ентропії, як спрощений випадок визначається кількість унікальних значень зміщених синаптичних сигналів.

Output Transformation (Activation) — реалізує оцінювання результату агрегації для прийняття рішення щодо формування вихідного сигналу, зокрема для дискретного перцептрона виконується перевірка на рівність із заданим значенням (A).

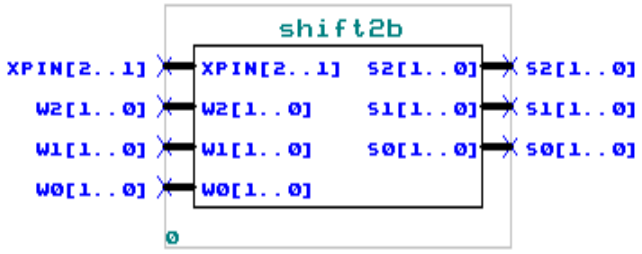
Таким чином, об’єктом дослідження є дискретна перцептронна структура, реалізована на основі ймовірнісного оцінювання зміщених синаптичних сигналів.

Предметом дослідження є методи та засоби апаратної реалізації перцептронних структур на основі програмованих логічних інтегральних схем (ПЛІС).

Метою дослідження є розробка, а також функціональне та інструментальне дослідження дискретного (бінарного) двовходового перцептрона для емуляції логічних булевих функцій, що реалізується на основі платформи програмованих логічних інтегральних схем.

Апаратна та програмна реалізації компонентів дискретного перцептрона

З огляду на подану вище структуру дискретного перцептрона (див. рис. 1) компонент Input Transformation, що реалізує зміщення вхідних бінарних синаптичних сигналів x_1, x_2 ($\overline{X}$) шляхом додавання відповідних коефіцієнтів w_0, w_1, w_2 ($\overline{W}$), нескладно реалізувати як паралельний суматор засобами мови AlteraHDL [10]—[12], (рис. 2).




```
SUBDESIGN Shift2b (
  xPin[2..1]:                  INPUT;
  w2[1..0],w1[1..0],w0[1..0]: INPUT;
  s2[1..0],s1[1..0],s0[1..0]: OUTPUT;
)
BEGIN
  s2[]=w2[]+(0,xPin[2]);
  s1[]=w1[]+(0,xPin[1]);
  s0[]=w0[];
END;
```

Рис. 2. Позначення та реалізація на AHDL компонента shift2b зміщення синаптичних сигналів на два бінарні входи

Структуру та опис апаратного компонента **shift2b**, який реалізує компонент Input Transformation для двох бінарних синаптичних сигналів показано на рис 2, символ у схемному редакторі (ліва частина), показує зовнішній інтерфейс модуля:

– вектори-входи xPin[2..1] містять бітові значення вхідних сигналів x_2, x_1;

– вектори-входи коефіцієнтів зсуву w2[1..0], w1[1..0], w0[1..0] задають наперед вибрані двобітні зміщення w_2, w_1, w_0;

– вектори-виходи s2[1..0], s1[1..0], s0[1..0] формують зміщені синаптичні сигнали, які надалі подаються на компонент Aggregation.

Фрагмент коду AHDL (права частина рис. 2), що описує логіку модуля, для кожного каналу формується двобітний вектор:

s2[] = w2[] + (0, xPin[2]); — MSB = 0, LSB = x2

s1[] = w1[] + (0, xPin[1]); — MSB = 0, LSB = x1

s0[] = w0[]; — постійний коефіцієнт зміщення

Операція «+» синтезується у паралельний двобітний суматор, тому зсув (додавання ваги) виконується апаратно за один такт. Проте відсутність операції додавання для ваги w0[] створює апаратну несиметричність між виходами s2[], s1[] та виходом s0[], що приводить до різних часових затримок формування згаданих вихідних сигналів. Цей факт необхідно враховувати у разі використання у асинхронних системах. Таким чином, shift2b компактно реалізує зміщення вхідних бітів на основі простого додавання, мінімізуючи апаратні ресурси ПЛІС і забезпечуючи готові, вирівняні за розрядністю вектори для подальшого ентропійного агрегування.

Комплексну перевірку компонента **shift2b** у двох площинах — функціональній та часовій показано на рис. 3.

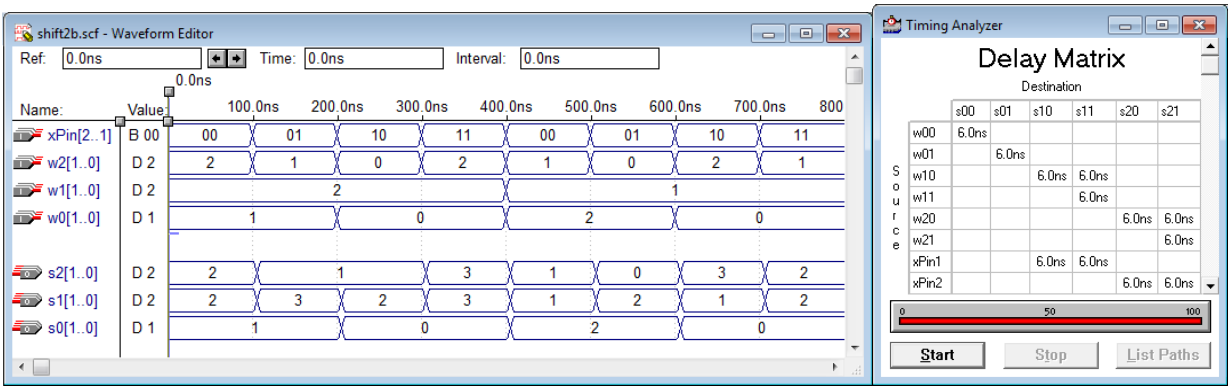


Рис. 3. Результати функціонального та інструментального моделювання компонента shift2b зміщення синаптичних сигналів на два бінарні входи

Осцилограма функціоналу в середовищі Waveform Editor (ліва частина), відображає реакцію виходів s2[1..0], s1[1..0], s0[1..0] на послідовне перебірне змінювання комбінацій вхідних бітів xPin[2..1] ("00", "01", "10", "11") за фіксованих значень ваг w2, w1, w0. На кожному такті можна побачити, що:

– s2 дорівнює двобітній сумі $w_2 + x_2$;

– s1 дорівнює $w_1 + x_1$;

– s0 незмінно повторює w_0.

Коректність логіки підтверджується тим, що для всіх 16 перевірених комбінацій входів отримані очікувані числові коди (зокрема, при "xPin=01" та "w1=2" на виході формується "s1=3").

Матриця затримок (права частина рис. 3) середовища Timing Analyzer показує для кожної пари «джерело → призначення» однорідну затримку 6 нс, що означає:

– усі шляхи від будь-якого входу (xPin[], w2[], w1[], w0[]) до будь-якого біта виходу (s2[], s1[], s0[]) мають однакову, невелику критичну затримку;

– модуль може надійно працювати з тактовою частотою близько $1/6\,\text{нс} \approx 166\,\text{МГц}$ без порушення часових обмежень.

Отже, функціональне моделювання підтвердило правильність арифметичного зміщення, а інструментальний аналіз показав рівномірну та прийнятну часову поведінку, через що **shift2b** є ефективним і масштабованим блоком для подальшого використання у дискретному перцептроні.

На подальшому етапі розробки необхідно реалізувати компонент формування результату (Output Transformation чи Activation), що є компаратором двобітових чисел s_1 та s_0. Фактично цей компонент буде також використано під час реалізації алгоритму визначення кількості унікальних значень зміщених синаптичних сигналів, що формує компонент **shift2b**. Позначення та реалізація засобами мови AlteraHDL [10-12], подано на рис.4.

Компактна апаратна реалізація компонента **cmp2b**, який виконує функцію Output Transformation (Activation) для дискретного перцептрона — двобітовий компаратор, що перевіряє рівність між зміщеними синаптичними сигналами s1[1..0] та s0[1..0] показана на рис. 4.

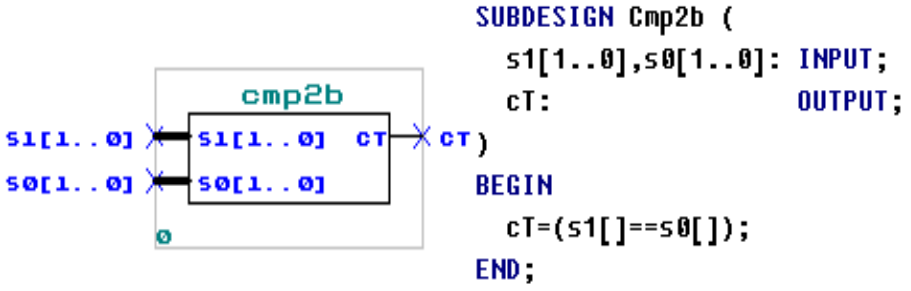


Рис. 4. Позначення та реалізація на AHDL компонента cmp2b (компаратора) перевірки на рівність двох двобітних чисел

Символ у схемному редакторі (ліва частина) показує зовнішній інтерфейс модуля, що складається з двох 2-бітових входів s1[1..0], s0[1..0] і одного однобітового виходу cT, який стає "1", коли обидва вектори тотожно рівні, та "0" — інакше. Саме цей біт інтерпретується як результат активації (y). Фрагмент коду AHDL (права частина рис. 4), мінімалістичний опис логіки модуля [10]—[12].

Єдина операція порівняння «==» синтезується інструментом у двобітний елемент "XNOR + AND", що використовує лише кілька логічних елементів ПЛІС і не вимагає тактування. Завдяки цьому компонент **cmp2b** додає незначну апаратну затримку до критичного шляху й може безпосередньо підключатися до подальших стадій агрегування або ансамблевої логіки кількості унікальних значень.

Таким чином, **cmp2b** можна ефективно використати розв’язуючи задачу компонента агрегації для порівнювання вхідних зміщуваних синаптичних сигналів s1[], s0[], а також як компонент активації, що завершує каскад "shift2b → cmp2b", забезпечуючи швидке та ресурсоефективне формування вихідного сигналу дискретного перцептрона.

Результати перевірки працездатності компонента **cmp2b** — функціональне моделювання в середовищі Waveform Editor та аналіз часових характеристик у Timing Analyzer показано на рис. 5.

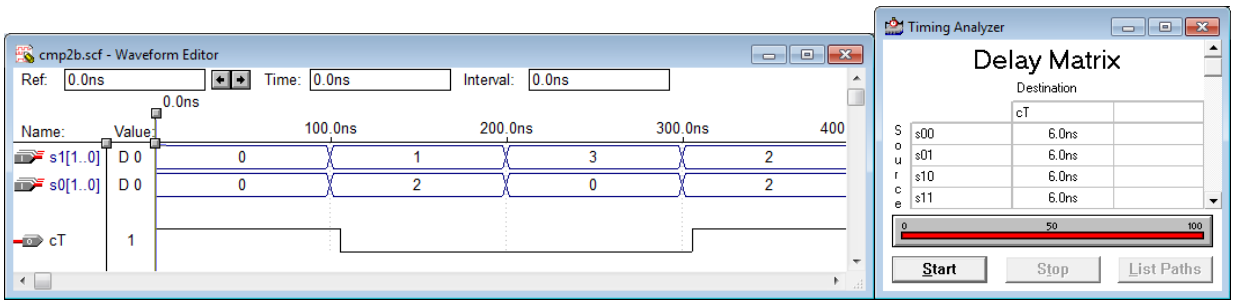


Рис. 5. Результати функціонального та інструментального моделювання компонента cmp2b (компаратора) перевірки на рівність двох двобітних чисел

Осцилограма функціоналу в середовищі (ліва частина) показує зміну двобітових вхідних векторів s1[] і s0[] протягом чотирьох тактових інтервалів (00–01–11–10). Вихід cT набуває логічної "1" лише тоді, коли обидва вектори збігаються ("00" на початку та "10" наприкінці), і залишається "0" для різних комбінацій ("01"та"10", "11"та"00" тощо), що підтверджує коректність логічної умови cT = (s1[] == s0[]).

Матриця затримок — Delay Matrix (права частина рис. 3) середовища Timing Analyzer показує для всіх можливих бітів джерела (s00, s01, s10, s11) до виходу cT зафіксовано однакову критичну затримку в 6 нс. Тобто, компаратор практично не створює суттєвих апаратних затримок й забезпечує стабільну роботу на частотах близько 166 МГц, що відповідає часовим параметрам попереднього блока **shift2b**.

Отже, функціональні та інструментальні результати підтверджують правильність і високошвидкісну роботу **cmp2b**, завершуючи ланцюжок логіки «зміщення → порівняння» у дискретному перцептроні.

Схемна реалізація компонента агрегування (Aggregation) ґрунтується на використанні результатів порівнювання зміщених синаптичних сигналів s_2, s_1, s_0, які є аргументами логічних функцій, що формують біти числа результату c_1 та c_0.

Таким чином, для формування бітів результату c_1 та c_0 компонент агрегування оперує лише трьома булевими ознаками — результатами попарного порівняння зміщених синаптичних сигналів: $c_{01} = (s_0 = s_1)$; $c_{02} = (s_0 = s_2)$; $c_{12} = (s_1 = s_2)$. Ці три ознаки повністю визначають кількість унікальних значень $U \in \{1, 2, 3\}$ серед $\{s_2, s_1, s_0\}$. Кількість унікальних станів U задається двома бітами c_1 та c_0 за правилом, поданим у табл. 2.

Таблиця 2

U (кількість унікальних станів)	Біти c_1c_0	Пояснення
1 (усі рівні)	01	$s_2 = s_1 = s_0$
2 (один відрізняється)	10	рівні лише двоє
3 (усі різні)	11	жодної рівності

Як можна побачити з табл. 2, c_1 набуває значення «1», якщо серед трьох порівнянь є хоча б одна нерівність, а c_0 дорівнює «1» у двох крайніх випадках — коли усі рівні або коли жодна пара не рівна. Таким чином, логічні функції для c_1 та c_0 можна записати у такому вигляді:

$$c_1 = \overline{c_{01} \wedge c_{02} \wedge c_{12}};\ c_0 = (c_{01} \wedge c_{02} \wedge c_{12}) \vee \overline{c_{01} \wedge c_{02} \wedge c_{12}}.$$

Отримані логічні функції дозволяють синтезувати компактний, однотактний вузол Aggregation (**cnts**), який кодує кількість унікальних значень серед $\{s_2, s_1, s_0\}$ у вигляді двобітового результату c_1c_0. Позначення та логічну схему компонента **cnts** — визначення кількості унікальних значень, показано на рис. 6.

Вузол Aggregation **cnts**, який об'єднує три двобітові входи s2[], s1[], s0[] і формує двобітовий код кількості унікальних значень c[], подано на рис. 6. Ліва частина — символ у схемному редакторі, що показує зовнішній інтерфейс модуля, який складається з трьох 2-бітових шин (зміщені синаптичні сигнали) та однієї 2-бітової шини результату c[], що відображає кількість унікальних станів, причому:

– c[1] — старший біт (сигналізує, чи є серед трійки хоча б одна нерівність);

– c[0] — молодший біт (розрізняє крайні випадки «усі рівні» та «усі різні»).

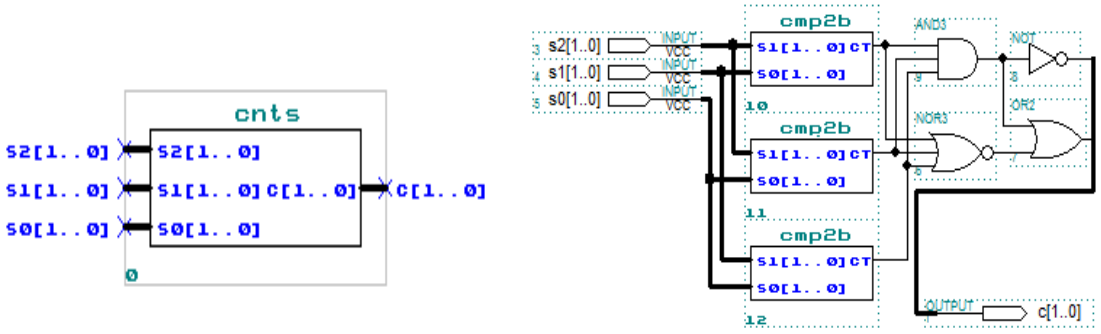


Рис. 6. Позначення та схемна реалізація компоненту cnts визначення кількості унікальних значень для трьох двобітних чисел

Структурна схема логіки (права частина рис. 6) складається з трьох модулів **cmp2b** (рис. 4) та логічної схеми для синтезу кількості унікальних станів. Модулі **cmp2b** попарно порівнюють:

– cmp2b_0 — s2 з s1 → вихід c12;

– cmp2b_1 — s2 з s0 → вихід c20;

– cmp2b_2 — s1 з s0 → вихід c10.

Логічна схема на виході реалізує $c_1 = \overline{c_{01} \wedge c_{02} \wedge c_{12}}$; $c_0 = (c_{01} \wedge c_{02} \wedge c_{12}) \vee \overline{c_{01} \vee c_{02} \vee c_{12}}$.

– тривходовий AND3 обчислює $c_{01} \wedge c_{02} \wedge c_{12}$;

– NOT інвертує цей сигнал для формування c_1;

– тривходовий NOR3 генерує $\overline{c_{01} \vee c_{02} \vee c_{12}}$;

– двовходовий OR2 об'єднує результати AND3 та NOR3, формуючи c_0.

Завдяки використанню вже синтезованих компараторів та чотирьох базових логічних елементів модуль **cnts** реалізує підрахунок унікальних значень за один такт, забезпечуючи узгоджену затримку з попередніми компонентами "shift2b → cmp2b" і готовий двобітовий код для подальшого розв'язання задачі перцептронної класифікації.

Комплексну перевірку модуля cnts, що підраховує кількість унікальних значень серед трьох двобітових сигналів подано на рис. 7.

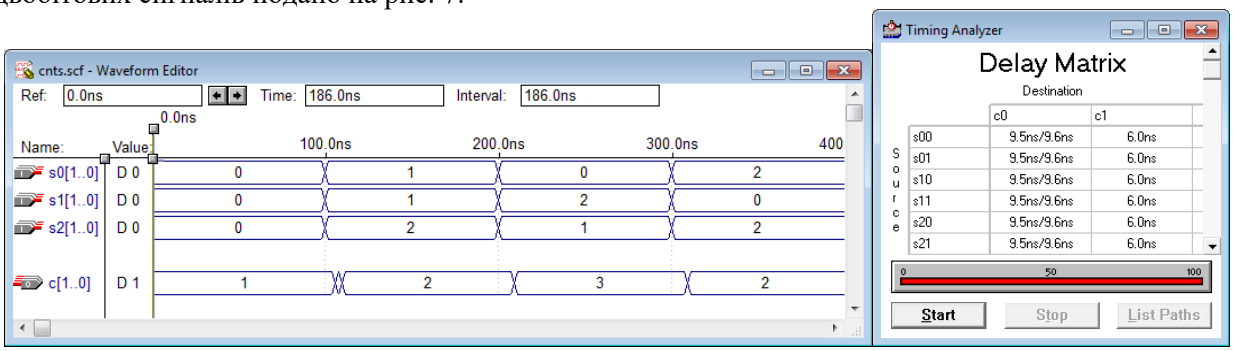


Рис. 7. Результати функціонального та інструментального моделювання компонента cnts — визначення кількості унікальних значень для трьох двобітних чисел

– Осцилограма функціоналу (ліва частина) де вхідні вектори s0[], s1[], s2[] перебирають комбінації, у яких:

– на 0 нс усі вхідні сигнали рівні ("00 00 00") ⇒ c[] = 01 (код "1 унікальне");

– на ≈100 нс лише s2 відрізняється ("01 01 10") ⇒ c[] = 10 (код "2 унікальні");

– на ≈200 нс усі різні ("00 10 01") ⇒ c[] = 11 (код "3 унікальні");

– на ≈300 нс лише s1 відрізняється ("10 00 10") ⇒ c[] = 10, повернення до випадку "2 унікальні". Отже, вихідний код c[] точно відповідає очікуваній кількості унікальних значень у кожному такті. Матриця затримок (права частина рис. 7) показує:

– для старшого біта c1 (формується інверсією AND3) критична затримка становить 6 нс для всіх маршрутів, що збігається з попередніми блоками **shift2b** та **cmp2b**.

– молодший біт c0 (складніша логіка AND3 ∨ NOR3) має дещо більшу затримку ≈9,5 нс.

Це гарантує роботу модуля **cnts** на тактових частотах близько105 МГц (обмежено шляхами до

c0), що залишається узгодженим з раніше синтезованими компонентами каскаду. Загалом, функціональне та часове моделювання підтверджують правильність кодування кількості унікальних значень і прийнятні швидкісні характеристики компонента агрегування **cnts**.

Використання попередньо реалізованих та досліджених компонентів **shift2b** (рис. 2), **cmp2b** (рис. 4) та **cnts** (рис. 6) дозволяє реалізувати дискретний перцептрон на два бінарні входи для емуляції логічних функцій. Схемна реалізація перцептрона, як інтеграція раніше спроєктованих блоків у єдиний модуль **perceptron**, показана на рис. 8. Схемне позначення модуля (ліва частина) показує мінімалістичний зовнішній інтерфейс:

– xb[] — вхідний вектор початкових бітів xb_2, xb_1;
– w2[], w1[], w0[] — двобітні коефіцієнти зміщення;
– aN[] — еталон-поріг (аналог A) для остаточного порівняння;
– y — однобітовий вихід перцептрона.

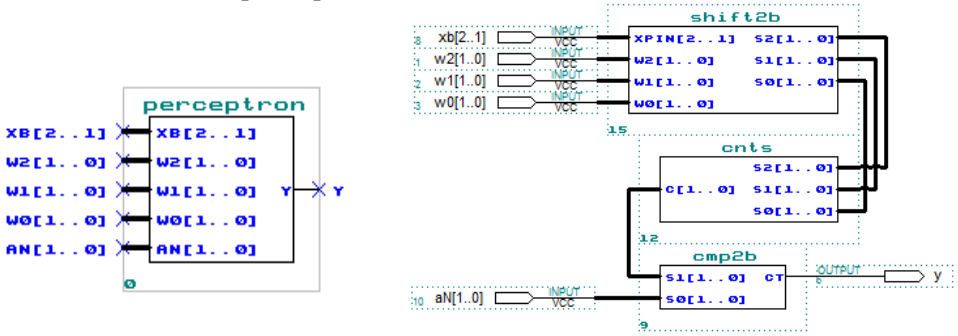


Рис. 8. Позначення та схемна реалізація дискретного перцептрона на два бінарні входи для емуляції логічних функцій

Структурна схема логіки (права частина рис. 8) показує, як дані послідовно проходять крізь внутрішні компоненти:

s – **hift2b** виконує паралельне додавання $x_i + w_i$, формуючи зміщені сигнали s2[], s1[], s0[];

– **cnts** визначає кількість унікальних значень серед s2[], s1[], s0[], фактично реалізуючи спрощений ентропійний критерій агрегації;

– **cmp2b** порівнює отриманий код з порогом aN[], результат порівняння на рівність транслюється безпосередньо на вихід (y).

Завдяки такій каскадній організації модуль **perceptron** зберігає однобітну комбінаційну затримку, успадковану від найповільнішого шляху (shift2b → cnts → cmp2b), та забезпечує гнучке налаштування логічної функції лише зміною коефіцієнтів w2[], w1[], w0[] і порогу aN[], без перепрошивки коду або структурних змін схеми.

Результати функціонального та інструментального моделювання (рис. 9) підтверджують, що інтегрований модуль **perceptron** коректно емулює шість базових логічних операцій, а також демонструє його часові обмеження на кристалі ПЛІС.

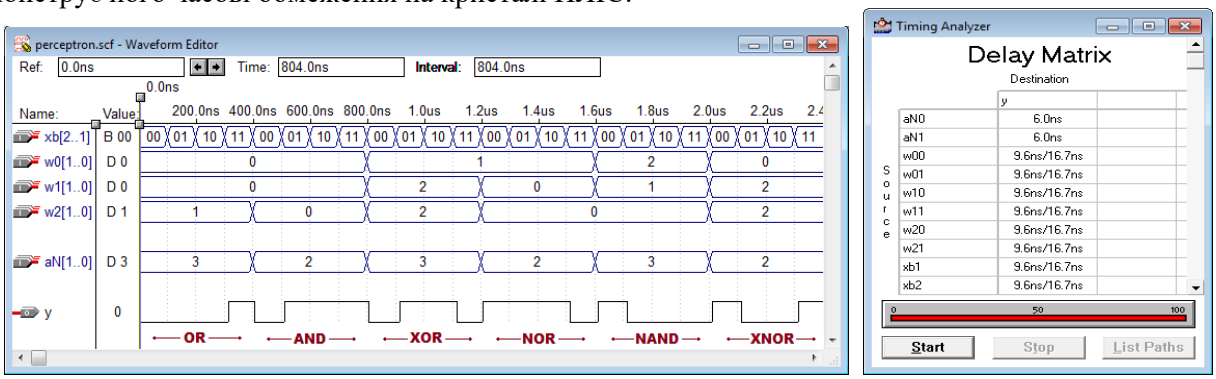


Рис. 9. Результати функціонального та інструментального моделювання дискретного перцептрона на два бінарні входи у разі емуляції логічних функцій or, and, xor та nor, nand, xnor

Осцилограма функціоналу (ліва частина) показує сигнали на входах xb[], що змінюються по циклу "00, 01, 10, 11" кожні 200 нс. Коефіцієнти зсуву w0[], w1[], w2[] і поріг aN[] залишаються постійними на інтервалі моделювання відповідної логічної функції переформовуються через 800 нс, задаючи послідовно пари «параметри → модельована функція», підписані внизу діаграми:

w0[] = w1[] = 0, w2[] = 1, aN[] = 3 → OR (y = 0 тільки при xb[] = 00).
w0[] = w1[] = w2[] = 0, aN[]= 2 → AND (y = 1 тільки при xb[] = 11).
w0[] =1, w1[] = w2[] = 2, aN[] = 3 → XOR.
w0[] = 1, w1[] = w2[] = 0, aN[] = 2 → NOR.
w0[] = 2, w1[] =1, w2[] = 0, aN[] = 3 → NAND.
w0[] =0, w1[] = w2[] = 2, aN[] = 2 → XNOR.

На кожному інтервалі вихід (y) реалізує саме ту таблицю істинності, яка очікується для відповідної логічної операції, підтверджуючи гнучкість налаштування перцептрона лише зміною коефіцієнтів і порога без перепроектування схеми. Матриця затримок (права частина рис. 9) показує, що шляхи від еталонних бітів порога aN0/aN1 до виходу мають критичну затримку 6 нс (одна операція NOT або AND→NOT). Шляхи від вагових та інформаційних входів (w0[], w1[], w2[], xb[]) довші: 9,6 нс (min) / 16,7 нс (max). Найповільніша ділянка проходить через ланцюг "shift2b → cnts → cmp2b". Фактично, найбільша затримка 16,7 нс задає верхню оцінку робочої тактової частоти ≈60 МГц для повністю комбінаційної реалізації. За потреби вищих швидкостей блок легко конвеєрується, оскільки між компонентами існують чітко визначені шини даних.

Таким чином, функціональне моделювання доводить коректність емуляції будь-яких бінарних логічних функцій через параметризацію ваг та порога, а інструментальний аналіз підтверджує, що підсумковий дискретний перцептрон відповідає вимогам швидкодії й може бути безпосередньо інтегрований у більші цифрові системи чи нейромережеві прискорювачі на ПЛІС.

Висновки

Дослідження й поетапне конструювання дискретного двовходового перцептрона на ПЛІС продемонстрували, що поєднання трьох мінімалістичних компонентів — shift2b (зміщення синаптичних сигналів), cnts (агрегація на основі кількості унікальних значень) та cmp2b (активаційний компаратор) — дозволяє побудувати однорівневий класифікатор з чітко визначеною геометрично-статистичною інтерпретацією. До ключових переваг згаданого підходу варто віднести:

– структурна простота та модульність. Кожний блок синтезується у 2—3 логічні елементи, що спрощує верифікацію, повторне використання та масштабування на більшу кількість входів;

– універсальність параметризації. Зміною трьох коефіцієнтів зміщення w2,w1,w0 та порогового значення aN перцептрон реалізує шість базових булевих операцій (OR, AND, XOR, NOR, NAND, XNOR) без перепрошивки топології;

– передбачувана швидкодія. Найбільша виміряна критична затримка 16,7 нс (≈ 60 МГц) узгоджена з вимогами вбудованих систем реального часу; за потреби продуктивність легко підвищується конвеєризацією.

До обмежень варто віднести лінійну природу ухвалення рішення: бінарний вихід формується лише шляхом перевірки рівності агрегованого коду з порогом. Це зумовлює:

– ймовірно обмежену виразність щодо нелінійно роздільних задач (зокрема, багатовимірний XOR поза досяжністю без каскадування кількох таких блоків);

– часткову чутливість до шуму вхідних сигналів, оскільки дискретні зсуви не враховують ступінь відхилення, а лише факт рівності/нерівності.

Разом з тим, отримані аналітичні оцінки компонентної складності (порівняння кількості елементів для функцій 1-го й 2-го порядків) показують, що перехід від класичного лінійного суматора до ентропійного (ймовірнісного) критерію cnts дає виграш у ресурсах для реалізації еквівалентних булевих функцій. Це підтверджує доцільність статистичних (ймовірнісних) підходів у вдосконаленні одношарових перцептронів.

В результаті, запропонована архітектура є ефективною основою для побудови легковагових цифрових нейромережевих компонентів, що здатні виконувати булеві обчислення й базові класифікаційні задачі в реальному часі з мінімальними апаратними затратами, водночас залишаючи відкритими перспективи для подальшого нарощування функціональності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] P. Bartoli, C. Veronesi, A. Giudici, D. Siorpaes, D. Trojaniello, and F. Zappa, “Benchmarking Energy and Latency in Tinyml: A Novel Method for Resource-Constrained AI,” *ArXiv*, 2025, 15622. https://doi.org/10.48550/arXiv.2505.15622 .

[2] C. Kachris, “A survey on hardware accelerators for large language models,” *Appl. Sci.*, vol. 15, no. 2, p. 586, 2025.

[3] Y. Zhu, “Analysis and application on enhancing CNN performance via FPGA integration,” in *Int. Conf. Electron. Elect. Inf. Eng.*, S. Li and B. Hu, Eds. Haikou, China, Aug. 16-18, 2024. SPIE, 2024, p. 31. https://doi.org/10.1117/12.3052318 .

[4] R. Appuswamy, et al., “Breakthrough Low-Latency, High-Energy-Efficiency LLM Inference Performance Using North-Pole,” in *2024 IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Wakefield, MA, USA, Sep. 23-27, 2024. IEEE, 2024, pp. 1-8. https://doi.org/10.1109/hpec62836.2024. 10938418 .

[5] A. Nechi, L. Groth, S. Mulhem, F. Merchant, R. Buchty, and M. Berekovic, “FPGA-based deep learning inference accelerators: Where are we standing?” *ACM Trans. Reconfigurable Technol. Syst.*, Sep. 2023. https://doi.org/10.1145/3613963 .

[6] J. Yik, et al., “The neurobench framework for benchmarking neuromorphic computing algorithms and systems,” *Nature Commun.*, vol. 16, no. 1, Feb. 2025. https://doi.org/10.1038/s41467-025-56739-4 .

[7] С. І. Мельничук, і С. В. Яковин, «Спосіб реалізації перцептрона на основі імовірнісних характеристик зміщених синаптичних сигналів,» *Патент України 126753*, січ. 25, 2023.

[8] S. Melnychuk, M. Kuz, and S. Yakovyn, “Emulation of logical functions NOT, AND, OR, and XOR with a perceptron implemented using an information entropy function,” in *2018 14th Int. Conf. Adv. Trends Radioelecrtronics, Telecommun. Comput. Eng. (TCSET)*, Lviv-Slavske, Ukraine, Feb. 20-24, 2018. IEEE, 2018. https://doi.org/10.1109/tcset.2018.8336337 .

[9] S. V. Yakovyn, and S. I. Melnychuk, “Discrete perceptron based on probabilistic estimates of shifted synaptic signals,” *Nauk. Visnyk Natsionalnoho Hirnychoho Universytetu*, no. 2, pp. 189-196, 2025. https://doi.org/10.33271/nvngu/2025-2/189 .

[10] A. Guesmi, I. Alouani, M. Baklouti, T. Frikha, and M. Abid, “SIT: Stochastic input transformation to defend against adversarial attacks on deep neural networks,” *IEEE Des. & Test*, vol. 39, pp. 63-72, 2022. https://doi.org/10.1109/mdat.2021.3077542 .

[11] A. R. Omondi, and J. C. Rajapakse, Eds., *FPGA Implementations of Neural Networks*. Springer US, 2006. https://doi.org/10.1007/0-387-28487-7 .

[12] A. Ananthakrishnan, and M. G. Allen, “All-Passive hardware implementation of multilayer perceptron classifiers,” *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1-10, 2020. https://doi.org/10.1109/tnnls.2020.3016901 .

Рекомендована кафедрою комп'ютерних систем управління ВНТУ

Стаття надійшла до редакції 6.06.2025

Яковин Сергій Васильович — аспірант кафедри комп’ютерних систем і мереж, e-mail: ysv_@ukr.net ;

Мельничук Степан Іванович — д-р техн. наук, професор, завідувач кафедри комп’ютерних систем і мереж, e-mail: stenni@ukr.net ;

Мануляк Ірина Зіновіївна — канд. техн. наук, доцент, доцент кафедри комп’ютерних систем і мереж, e-mail: manulyak-iryna@ukr.net .

Івано-Франківський технічний університет нафти і газу, м. Івано-Франківськ

S. V. Yakovyn[1]
S. I. Melnychuk[1]
I. Z. Manuliak[1]

Implementation of a Two-Input Discrete Perceptron with Shifted Synaptic Signals on FPGA Using AlteraHDL

[1]Ivano-Frankivsk National Technical University of Oil and Gas

The paper proposes and experimentally investigates a hardware implementation of a discrete two-input probabilistic perceptron on a Field-Programmable Gate Array (FPGA). The perceptron is constructed from three elementary modules — shift2b (synaptic signal shifting via simple addition), cnts (aggregation based on counting the number of unique values), and cmp2b (a two-bit activation comparator). The hardware implementation of the discrete perceptron relies on shifting input signals using an integer addition operation, which significantly reduces hardware requirements.

Additionally, the proposed perceptron architecture ensures minimal component complexity (2–3 logic elements per block) and enables, by merely altering the weights and threshold, the emulation of six basic Boolean operations — OR, AND, XOR, NOR, NAND, and XNOR. This approach enables the creation of hardware mono-structural components based on a unified block, capable of implementing different logical functions depending on application requirements.

Functional simulation confirmed the correctness of all implemented truth tables, and timing analysis indicated a critical path delay of 16.7 ns, corresponding to an operating frequency of approximately 60 MHz without pipelining. The derived analytical relations demonstrate the potential for reducing hardware resource usage compared to traditional linear adders when synthesizing first- and second-order logic functions.

The proposed approach paves the way for scaling to a higher number of inputs, integration of statistical (probabilistic) aggregation criteria, and the development of embedded on-chip learning procedures. The results confirm the viability of using discrete perceptron structures as lightweight, energy-efficient classifiers in real-time systems and specialized neural network accelerators.

Keywords: perceptron, binary signals, Boolean functions, signal processing, field-programmable gate arrays (FPGA), neural networks.

Yakovyn Serhiy V. — Post-Graduate Student of the Chair of Computer Systems and Networks, e-mail: ysv_@ukr.net;

Melnychuk Stepan I. — Dr. Sc. (Eng.), Professor, Head of the Chair of Computer Systems and Networks, e-mail: stenni@ukr.net ;

Manuliak Iryna Z. — Cand. Sc. (Eng.), Associate Professor, Associate Professorof the Chair of Computer Systems and Networks, e-mail: manulyak-iryna@ukr.net