УДК 681.515+62 — 551.453

*Кондратенко Ю. П., Сидоренко С. А.*

# МЕТОДИ ПРОЕКТУВАННЯ НЕЧІТКИХ ПРИСТРОЇВ ПРИЙНЯТТЯ РІШЕНЬ НА ОСНОВІ ПРОГРАМОВАНИХ ЛОГІЧНИХ ІМС

*У статті аналізуються методи проектування нечітких пристроїв прийняття рішень та можливості використання програмованих логічних ІМС як елементної бази для побудови спеціалізованих фаззі-процесорів. Наведено порівняльний аналіз FPGA та CPLD-технологій ПЛІС. Детально описані підходи, що базуються на застосуванні спеціалізованих мов програмування ABEL, Verilog та VHDL. Запропоновано метод проектування універсальних НППР на основі пакета Active-HDL 3.5, наводяться структури окремих блоків НППР та VHDL-програми для їх реалізації. Результати моделювання поведінки НППР підтверджують ефективність застосування програмованих логічних ІМС для автоматизації процесів прийняття рішень в умовах невизна ченості.*

Необхідність прийняття рішень в умовах невизначеності призводить до застосування методів оптимізації та інтелектуальних алгоритмів і технологій при проектуванні систем підтримки прийняття рішень (СППР). Основними елементами таких систем є сенсорні пристрої, перетворювачі сигналів, пристрої цифрової обробки сигналів та пристрої прийняття рішень. Пристрій прийняття рішень є головним елементом будь-якої СППР, що визначає ефективність автоматизованого прийняття рішень в умовах невизначеності. Невизначеність умов функціонування об'єктів управління, а в деяких випадках і неможливість їх точної математичної формалізації, спонукає до необхідності застосування алгоритмів фаззі-управління для синтезу пристроїв прийняття рішень, що використовують лінгвістичні величини і висловлювання для опису стратегій прийняття рішень [1—4, 6, 7, 9, 11].

У запропонованому дослідженні автори аналізують можливі шляхи апаратної реалізації нечітких пристроїв прийняття рішень (НППР) на базі різноманітних платформ. Головна увага приділена підходу до проектування НППР, що базується на застосуванні програмованих логічних інтегральних схем (ПЛІС), як найсучаснішої елементної бази електронних пристроїв.

## 1. Загальна характеристика програмованої логіки

### *1.1. Сучасний стан ринку мікросхем програмованої логіки*

Програмована логіка є однією з найбільш динамічних галузей ринку електроніки. За статистичними даними провідних фірм Aldec Inc., США та Xilinx Inc., США (рис. 1, *а)* обсяг виробництва програмованих логічних інтегральних мікросхем (ІМС) з 1990 по 1998 роки зріс у понад 10 разів і продовжує зростати (лише в першому півріччі 1999 р. обсяг виробництва становив 662 млн USD, тоді як за весь 1998 р.—613 млн USD).

Сфера застосування програмованих логічних ІМС постійно розширюється. Діаграма розподілу обсягів споживання ПЛІС в окремих галузях наведена на рис. 1, *б*. Найбільшим споживачем ПЛІС є галузь телекомунікацій і зв'язку (39 % від всього обсягу виробництва). Друге місце обіймає галузь комп'ютерних мереж, що використовує 26 % обсягу виробництва ПЛІС. Крім того, програмовані логічні ІМС застосовуються в галузі цифрової обробки даних (19 %) та в індустріальному виробництві (16 %). Серед споживачів цих мікросхем можна назвати такі відомі фірми і концерни, як Alcatel, IBM, Booing, Lockheed, Hewlett Packard, Fujitsu, Hitachi, Silicon Graphics, Texas Instruments, Motorola, Rockwell, Kodak та багато інших.

До основних характеристик програмованих логічних ІМС належать [5]:

— кількість системних вентилів;
— швидкодія;
— системна тактова частота;
— максимальне число повторів вводу-виводу;
— енергоспоживання.

Сьогодні ПЛІС проектують і виробляють кілька десятків провідних фірм. Лідером у цій галузі є фірма Xilinx (обсяг виробництва тільки у 1-му півріччі 1999 р. становив 211,4 млн USD).

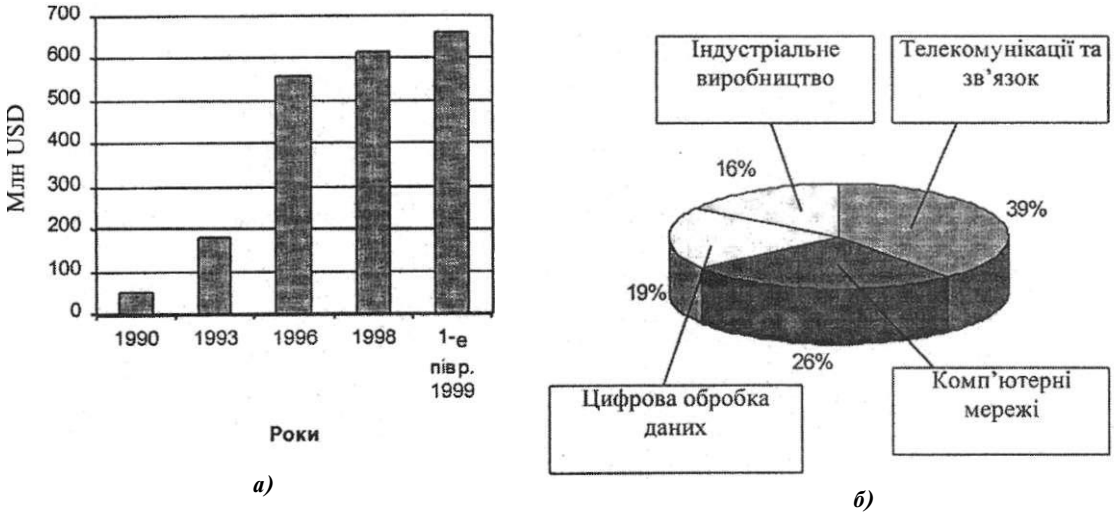



**Рис. 1.** Статистичні показники ринку програмованих ІМС
*а)* діаграма росту виробництва ПЛІС; *б)* обсяги застосування ПЛІС по окремих галузях

Фірма Xilinx спеціалізується на виробництві ІМС високої якості, зокрема, стійких до впливу радіації та інших зовнішніх збурень, що обумовлює широке застосування ПЛІС в аерокосмічній галузі, воєнній техніці та в промисловому виробництві. Програмовані логічні ІМС фірми Altera (197,8 млн USD) займають нішу масових ПЛІС невисокої собівартості. Фірма Lattice(59,7 млн USD) переважно розробляє перепрограмовані мікросхеми CPLD і утримує лідируючі позиції в цій галузі. Крім того, до основних виробників ПЛІС слід віднести фірму Actel (41,6 млн USD).

Серед номенклатури ПЛІС найбільшого поширення набули мікросхеми, що виготовлені за технологіями FPGA та CPLD. Розглянемо ці технології детальніше.

### *1.2. FPGA-технологія*

Абревіатура FPGA розшифровується як Field Programmable Gate Array — програмована користувачем вентильна матриця. Загальну структуру кристала FPGA-мікросхеми [5] наведено нарис. 2. По периферії верхнього шару кристала розміщуються блоки вводу/виводу (БВВ), що можуть бути запрограмовані для виконання функцій буферів: вхідного, вихідного, з запам'ятовуванням та ін. У деяких серіях FPGA-ІМС рівень напруги на двох БВВ може відрізнятись, що дає змогу зв'язувати різні за живленням інтерфейси.

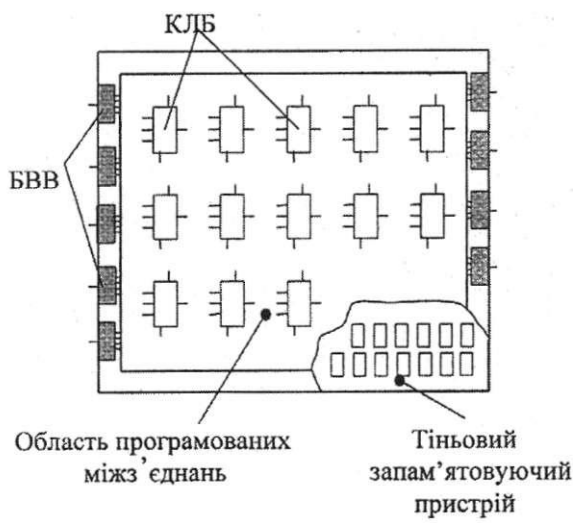



**Рис. 2.** Узагальнена структура FPGA-мікросхеми

У центрі кристала у вигляді матриці розміщено конфігуровані логічні блоки (КЛБ). Швидкодія мікросхем визначається часовою затримкою "вхід-вихід" одного КЛБ. Структура КЛБ залежить від серії мікросхем, наприклад, в ІМС XC2000 кожен КЛБ має 2 виходи, 4 входи загального призначення, спеціальний вхід синхронізації (тактовий вхід) та запам'ятовуючий елемент. КЛБ ІМС цієї серії може генерувати будь-яку логічну функцію чотирьох змінних або дві логічні функції трьох змінних. Змінні для логічних функцій можуть надходити з чотирьох входів та виходу запам'ятовуючого елемента.

Область між конфігурованими логічними блоками називається областю програмованих з'єднань [5] і являє собою розвинену ієрархію металічних ліній зв'язку, в місцях перетину яких розміщено спеціальні швидкодіючі транзистори. Функція області міжз'єднань полягає в забезпеченні зв'язку між будь-якими виводами КЛБ та БВВ. Потрібний маршрут міжблокових з'єднань в FPGA-ПЛІС реалізується комутацією відповідних ліній за допомогою транзисторів.

Нижній шар кристала займає тіньовий запам'ятовуючий пристрій, інформація в елементах якого і визначає логічні функції КЛБ, конфігурацію БВВ та маршрути міжз'єднань.

*Таблиця 1.* Основні характеристики мікросхем FPGA серій Virtex та Spartan

| Характеристика | Серія Virtex | | Серія Spartan | | |
|---|---|---|---|---|---|
| | XCV50 | XCV1000 | XCS40/XL | XCS20/XL | XCS05/XL |
| Системна частота (MHz) | 200 | 200 | 80 | 80 | 80 |
| Технологія (мкм) | 0.22 | 0.22 | 0.35/0.5 | 0.35/0.5 | 0.35/0.5 |
| Напруга живлення ядра (В) | 2.5 | 2.5 | 3.3/5.5 | 3.3/5.5 | 3.3/5.5 |
| Кількість системних вентилів (шт.) | 57906 | 1124022 | 13K-40K | 7K-20K | 2K-5K |
| Кількість логічних комірок (шт.) | 1728 | 27648 | 1862 | 950 | 238 |
| Макс. число портів вводу/виводу (шт.) | 180 | 514 | 205 | 160 | 77 |

У таблиці 1 наведено основні характеристики мікросхем FPGA однієї з останніх розробок серії Virtex та масової серії Spartan виробництва Xilinx Inc.

Широкий діапазон IMC FPGA-технології дозволяє проектувати на їх основі широкий спектр електронних пристроїв, серед яких: засоби поєднання різних за живленням інтерфейсів, перетворювачі кодів, периферійні контролери, мікропрограмні пристрої керування, скінченні автомати, універсальні та спеціалізовані процесори, пристрої цифрової обробки сигналів тощо.

### *1.3. CPLD-технологія*

Архітектура IMC типу CPLD (Complex Programmable Logic Device — комплексний програмований логічний пристрій) [5] зображена на рис. 3 (для прикладу взято архітектуру популярної серії мікросхем XC9500).

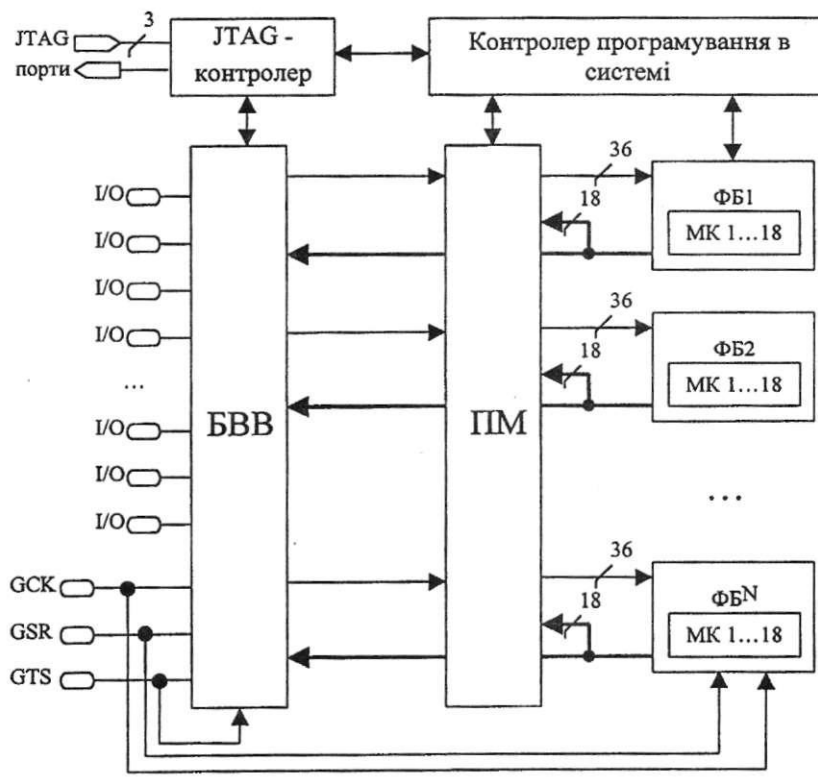



Рис. 3. Узагальнена структура CPLD-мікросхеми

*Таблиця 2.* **Основні характеристики ряду CPLD-мікросхем фірми Xilinx**

| Характеристика | Серія CoolRuner | | Серія XC9500 | |
|---|---|---|---|---|
| | XCR3320 | XCR22V10 | XC9536 | XC952288 |
| Кількість макрокомірок (шт.) | 320 | 10 | 36 | 288 |
| Системна частота (MHz) | 100 | 111 | 100 | 56.6 |
| Напруга живлення (В) | 5; 3 | 5; 3 | 5; 3 | 5; 3 |
| Кількість циклів перепрограмування | 1000 | 1000 | 10000 | 10000 |
| Час "pin-to-pin" (нс) | 7.5 | 7.5 | 5 | 15 |

IMC XC9500 має три групи виводів:

1) виводи JTAG-порта (стандарт IEEE Std. 1149.1 [10]) для програмування та периферійного сканування IMC;

2) порти вводу/виводу (I/O);

3) керуючі виводи: сигнал тактування GCK, установлення/скидання GSR, керування третім станом GTS.

Блоки вводу/виводу забезпечують буферизацію всіх входів та виходів IMC Кожен функціональний блок ФБ містить 18 макрокомірок із структурою "36 входів — 1 вихід" і дозволяє формувати 18 логічних функцій для будь-якої комбінації з 36 змінних. Матриця перемикань забезпечує подавання будь-яких вхідних сигналів та вихідних сигналів ФБ на входи ФБ, а також подавання вихідних сигналів ФБ на блоки вводу/виводу.

Основні характеристики ряду CPLD-мікросхем фірми Xilinx наведено в таблиці 2.

Програмовані логічні IMC CPLD-технології широко застосовуються для проектування нестандартних арифметико-логічних пристроїв, дешифраторів, мультиплексорів та ін.

До недоліків CPLD (у порівнянні з FPGA) слід віднести: малу кількість системних вентилів та значне енергоспоживання. Переваги їх полягають у вищій швидкодії та забезпеченні можливості встановлення захисту від копіювання. Важливою перевагою також є те, що програмні засоби для розробки та синтезу систем на базі CPLD поширюються вільно.

## 2. Методи проектування ПЛІС

Суть проектування ПЛІС полягає в розробці спеціальної програми (у вигляді двійкового потоку "bit-stream"), при завантаженні якої в IMC остання виконує функції спроектованого електронного пристрою.

### *2.1. Проектування на основі булевих рівнянь та схемне проектування*

Перший підхід до розробки ПЛІС ґрунтувався на проектуванні за допомогою булевих рівнянь. Такий спосіб проектування передбачає формування булевих рівнянь чи таблиць істинності для кожного тригера або блока вентилів. Опис проектних схем за допомогою булевих рівнянь є прийнятним для малих проектів, що включають десятки або навіть сотні вентилів, але для схем, що базуються на тисячах вентилів, такий опис стає надто громіздким і неефективним [8]. Крім того, схема, що формалізована за допомогою булевих рівнянь, важко піддається декомпозиції.

У традиційній формі схемне проектування є модифікацією методу проектування булевими рівняннями. Цей підхід ґрунтується на застосуванні попередньо синтезованих логічних блоків, таких, як регістри, лічильники, дешифратори та ін. Довгий час метод схемного проектування вважався найкращим вибором, однак з розвитком елементної бази мікросхемотехніки кількість елементарних схем у нових IMC постійно зростала. Разом з тим межею для розуміння проекту вважається 6000 схем [8], при перевищенні цієї межі застосування схемного проектування стає неефективним.

### *2.2. Застосування спеціалізованих мов для опису ПЛІС*

Найсучасніший підхід до проектування IMC ґрунтується на застосуванні спеціалізованих мов опису обладнання (HDL — Hardware Description Language). Ці мови, крім звичних конструкцій, таких, як розгалуження, цикли, підпрограми тощо, мають також спеціалізовані засоби — робота з портами, забезпечення паралельності виконання процесів тощо. Най-

поширенішими серед HDL стали мови ABEL, Verilog та VHDL [8].

ABEL (Advanced Boolean Equation Language — розширена мова логічних рівнянь) є промисловим стандартом, розробленим Data I/O Corp. для програмованих логічних пристроїв. ABEL може застосовуватися для опису поведінки систем (за допомогою С-подібних операторів) у різних формах: на основі логічних рівнянь, таблиць істинності, діаграм стану.

Порівняно з ABEL мови VHDL та Verilog є складнішими, але більш придатними для опису великих систем. Вони майже рівні за своїми технічними можливостями. В Європі та США ширше застосовується VHDL, в країнах Азії — Verilog.

VHDL базується на мові високого рівня Ada. З цієї мови розробниками VHDL було запозичено синтаксис та основні структури. Технологію розробки електронних компонентів за допомогою VHDL розглянемо на прикладі 8-бітового лічильника. Кожен об'єкт та його інтерфейс повинен бути описаний у розділі **entity.** Поведінка об'єкта визначається його архітектурою **(architecture),** архітектура складається з процесів **(process),** що виконуються паралельно. В наведеному прикладі описується об'єкт Counter, який містить 3 вхідних порти (CLK, RESET та ENABLE) і один 8-розрядний вихідний порт Q. Поведінка лічильника описується одним процесом. Після ключового слова **process** у дужках вказується список чутливості, тобто перелік портів, виникнення подій на яких викличе запуск процесу.

## 3. Системи автоматизованого проектування ПЛІС

### *3.1. Загальний огляд програмних засобів*

Відповідно до сучасних вимог системи автоматизованого проектування програмованих логічних ІМС повинні забезпечувати:

• реалізацію однієї чи більше HDL-мов З можливістю введення, редагування та відлагодження початкового тексту програм;

• реалізацію засобів графічного введення проектної схеми, наприклад, за допомогою редактора скінченних автоматів та засобів компіляції графічного представлення в HDL-код;

• реалізацію засобів моделювання поведінки описаного об'єкта;

• реалізацію засобів синтезу бітового потоку з підтримкою широкого класу серій ІМС;

• реалізацію засобів моделювання об'єкта на рівні вентилів;

• реалізацію засобів програмування ІМС

Розробкою такого програмного забезпечення займається значна кількість фірм, серед яких можна назвати Aldec, Xilinx, Viewlogic, Mentor, Synopsis, Vantage та ін.

Фірмою Xilinx розроблено кілька пакетів для автоматизованого проектування ПЛІС власного виробництва — серія Foundation та серія Alliance. їхні базові версії мають такі можливості: підтримка всіх типів CPLD; підтримка всіх FPGA-мікросхем ємністю до 20 000 системних вентилів; підтримка всіх FPGA-мікросхем серії SPARTAN; синтез логічними рівняннями; схемні проектування; мова опису обладнання ABEL; моделювання на рівні вентилів. Додатково САПР можуть комплектуватися засобами підтримки всіх типів FPGA-мікросхем, включаючи Virtex; засобами проектування, синтезу та оптимізації ПЛІС на базі мов VHDL та Verilog; інтерактивним довідником з мови VHDL тощо. Основним обмеженням цих програмних пакетів є те, що вони не мають засобів синтезу для мікросхем інших виробників ПЛІС.

Крім того, розроблено ряд програмних продуктів, що реалізують ту чи іншу частину загальних вимог до САПР ПЛІС. Застосування цих продуктів вимагає в кожному конкретному випадку розв'язання проблем сумісності між пакетами, тому перевагу слід надавати пакетам, що реалізують процес проектування ПЛІС

```
entity COUNTER is
  port (CLK    :in  STD_LOGIC;
        RESET  :in  STD_LOGIC;
        ENABLE :in  STD_LOGIC;
        Q      :out STD_LOGIC_VECTOR (7 downto 0));
end entity COUNTER;

architecture COUNTER_BEHAV of COUNTER is
begin
  process (CLK, RESET)
    variable QINT: STD_LOGIC_VECTOR (7 downto 0);
  begin
    if RESET='1' then
      QINT:="00000000";
    else
     if FALLING_EDGE(CLK) then
            if ENABLE='1' then
                 if QINT<255 then
              QINT:=QINT+1;
            else
              QINT:="00000000";
            end if;
            end if;
     end if;
    end if;
    Q<=QINT;
  end process;
end architecture COUNTER_BEHAV;
```

Рис. 4. VHDL-програма для реалізації 8-бітового лічильника

якнайповніше. Одним з таких пакетів є Active-HDL3.6 фірми Aldec Inc.

### *3.2. САПР Active-HDL 3.6 фірми Aldec Inc.*

Одним з лідерів у розробці програмного забезпечення САПР ПЛІС є компанія Aldec Inc. Остання версія пакета Active-HDL 3.6 вийшла в листопаді 1999 р. Завдяки політиці керівництва компанії, яка надає науково-технічну підтримку українським університетам, автори мають можливість розробляти проектні схеми за допомогою повної ліцензійної версії САПР Active-HDL.

Цей пакет включає широкий спектр засобів для проектування програмованих логічних ІМС [8]. Серед них:

• Design Browser — сучасний менеджер для керування проектом;

• Advanced HDL Editor — розвинений HDL редактор;

• Block Diagram Editor — засоби структурного проектування;

• State Machine Editor — редактор скінченних автоматів;

• Waveform Viewer/Editor — засоби для перегляду та редагування форм вхідних і вихідних сигналів;

• TCL/TK Scripting — реалізація мови сценаріїв TCL;

• Testbench Generation — генератор випробувальних стендів для тестування проектів;

• Behavioral Simulation — засоби моделювання поведінки проектних схем;

• RTL Simulation (Vital / SDF and EDIF) — засоби моделювання на рівні RTL (регістрових передач).

Крім того, на відміну від попередніх версій, в пакет включено засоби підтримки мови Verilog. Значним кроком вперед у розробці САПР ПЛІС є створені фірмою Aldec засоби для моделювання з підвищеною швидкодією CBS (Cycle Based Simulation). Як видно з діаграми на рис.5, витрати часу на моделювання проекту порівняно з моделюванням без CBS знижуються майже в 10 разів.

Затрати часу на моделювання є найбільш значною складовою загальних витрат часу при проектуванні програмованих логічних ІМС, тому підвищення швидкодії засобів моделювання (рис. 5) є одним з найперспективніших шляхів скорочення термінів проектування ІМС

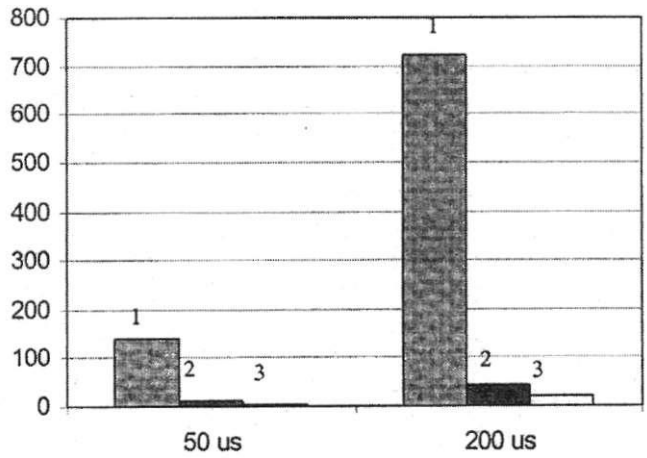
## 4. Проектування нечітких пристроїв прийняття рішень на основі FPGA-технології

### *4.1. Алгоритми роботи НППР*

Стратегія прийняття рішень у НППР задається за допомогою матриці знань (бази правил), рядками якої є лінгвістичні правила, що описують поведінку нечіткої системи [1,2,11]. Логічні оператори можуть поєднуватись як через операцію **І,** так і через операцію АБО, наприклад:

ЯКЩО (Похибка = "Висока Позитивна") **І** (Швидкість = "Низька")

ТО Управління := "Високе Негативне"




**Проект:** Rotator

**Число вентилів:** 9467

**Число тригерів:** 801

Режим 1 - Моделювання без CBS

Режим 2 - Моделювання з CBS (on-line)

Режим 3 - Моделювання з CBS (batch)

| Час моделювання | Режим 1 | Режим 2 | Режим 3 |
|---|---|---|---|
| 50 us | 142 с | 12 с | 5.2 с |
| 200 us | 723 с | 41 с | 17. 7 с |

Рис. 5. Гістограми витрат часу на моделювання проектних схем в САПР Active-HDL

Лінгвістичні терми ("Високий", "Низький" тощо) визначаються на спеціальній шкалі, один з варіантів якої показано на рис. 6.

Блок-схему алгоритму роботи НППР представлено на рис. 7 [3].

### *4.2. Аналіз можливих шляхів реалізації НППР*

Існує кілька можливих шляхів апаратної реалізації НППР. У праці [7] розглядаються три підходи: програмний (реалізація на універсальних контролерах, що не мають у своїй програмній моделі спеціальних команд маніпулювання з нечіткими множинами), програмно-апаратний (реалізація на спеціалізованих контролерах, що мають основний набір команд для виконання операцій нечіткої логіки), апаратний (базується на принципі поелементної декомпозиції: фазифікація, дефазифікація, обчислення істинності правил та ін.). Реалізація НППР на базі ПЛІС сприятиме поєднанню гнучкості програмного підходу із швидкодією апаратного підходу.

Архітектура ІМС НППР може бути побудована за двома принципами: проектування НППР "для конкретної задачі" при визначених форматах вхідних/вихідних координат, формах та коефіцієнтах функцій належності, компонентах матриці знань та у вигляді спеціалізованого фаззі-процесора. Перший підхід дозволяє досягти максимальної швидкодії НППР через максимальну паралелізацію процесів. Однак застосування цього підходу обмежується тим, що порівняно з фаззі-процесором він вимагає більше часу на розробку та потребує більшого числа системних вентилів. Другий підхід дає змогу скоротити час при перепроектуванні НППР (на розв'язання іншої задачі) тільки за рахунок можливості заміни частини VHDL-коду, яка визначає матрицю знань та інформацію про вхідні/вихідні координати, без необхідності внесення структурних змін у проектну схему НППР.

Розглянемо другий підхід детальніше.

### *4.3. Структура фаззі-процесора*

Структурна схема НППР як спеціалізованого фаззі-процесора для випадку з однією вихідною координатою зображена на рис. 8. Робота його здійснюється за таким алгоритмом.

1. Сигнал RST установлює показання системних лічильників RuleCounter та VarCounter"0", а також обнуляє масив накопичення.

2. Показання лічильників подаються на входи VarNumb та RuleNumb блоку зберігання матриці знань KnowMatrix.

3. На виходах MFType, C1...C5, Log, MFOutType, D1..D5 блоку KnowMatrix встановлюються відповідно: тип функції належності mfRuleNumb, VarNumb], значення її змінних коефіцієнтів, тип логічної функції поточного правила ("0" — AND, "1" — OR)$^5$THn функції належності вихідної змінної та її змінні коефіцієнти.

4. Блок MF_Lib (бібліотека функцій належності) здійснює обчислення значення функції належності типу MFType при вибраних значеннях змінних коефіцієнтів C1...C5 і значенні вхідної змінної X (що передається від блоку вибору вхідної змінної Sel_Var).

5. Блоки Min і Max, що за сигналом NextRule встановлюються відповідно в стани "1" і "0", здійснюють накопичення значень істинності для кожного стовпця матриці знань. Разом з тим у кожному циклі блок Min генерує сигнал Next Var.

6. За сигналом NextVar, якщо VarNumb < N (число вхідних координат), то системний лічильник VarNumb збільшує своє значення на 1, а керування передається на блок KnowMatrix (згідно з п. 3). Інакше (VarNumb - N) лічильник обнуляється і генерує сигнал End_Rule, який запускає процес накопичення.

7. Мультиплексор MP1 відповідно до сигналу Log перемикає на вихід Truth (значення

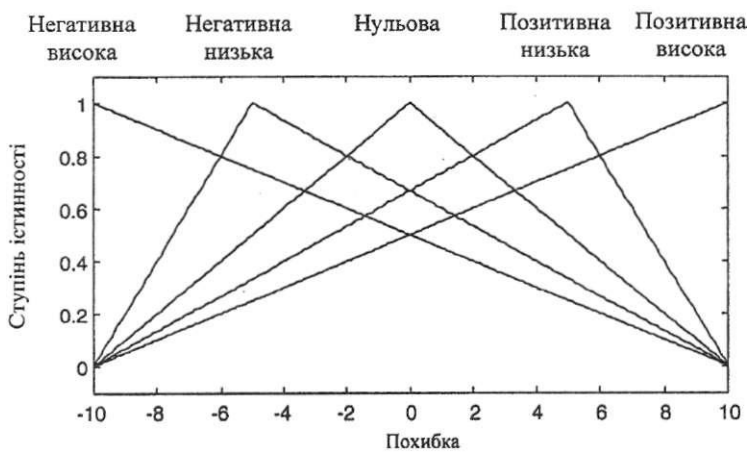



Рис. 6. Шкала лінгвістичних термів для змінної *Похибка*

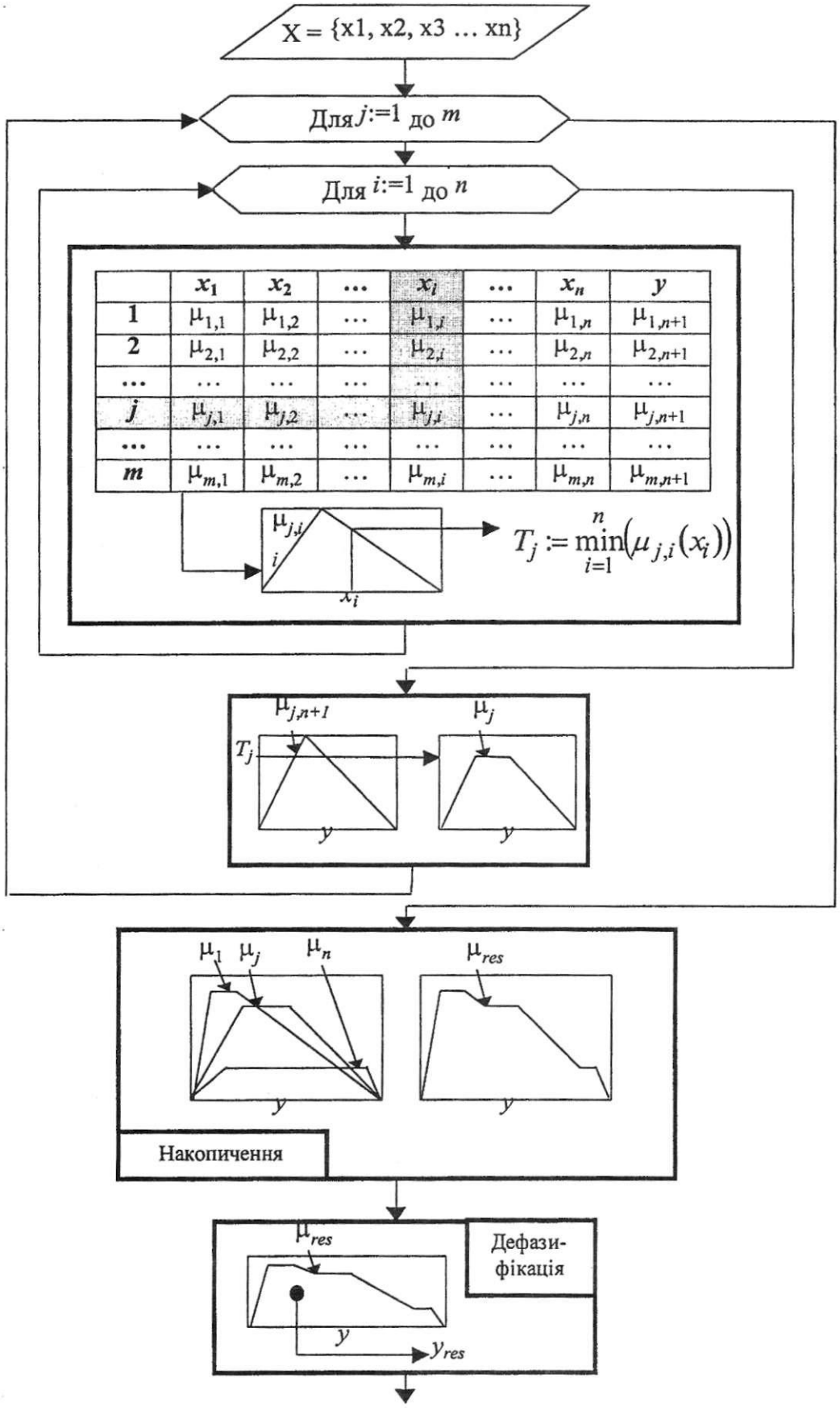



**Рис. 7.** Блок-схема алгоритму нечіткого прийняття рішень

істинності правила RuleNumb) сигнал від Min чи Max.

8. Блок накопичення Agregation обчислює вихідну функцію належності для правила номер RuleNumb і генерує сигнал NextRule.

9. Якщо RuleNumb < M (число правил), то системний лічильник RuleNumb збільшує своє значення на і, керування переходить на блок KnowMatrix (згідно з п. 3). Інакше (RuleNumb = = M) лічильник обнуляється і генерує сигнал End_ Matrix, який запускає процес дефаззіфікації.

10. Блок дефаззіфікації (Defuzzification) здійснює дефаззіфікацію вихідної координати методом обчислення центра ваги [11], надсилає результат на вихід IMC і генерує сигнал RST.

Застосування VHDL-технології до проектування спеціалізованого фаззі-процесора розглянемо на прикладі таких складових блоків НППР, як MF Lib та KnowMatrix.

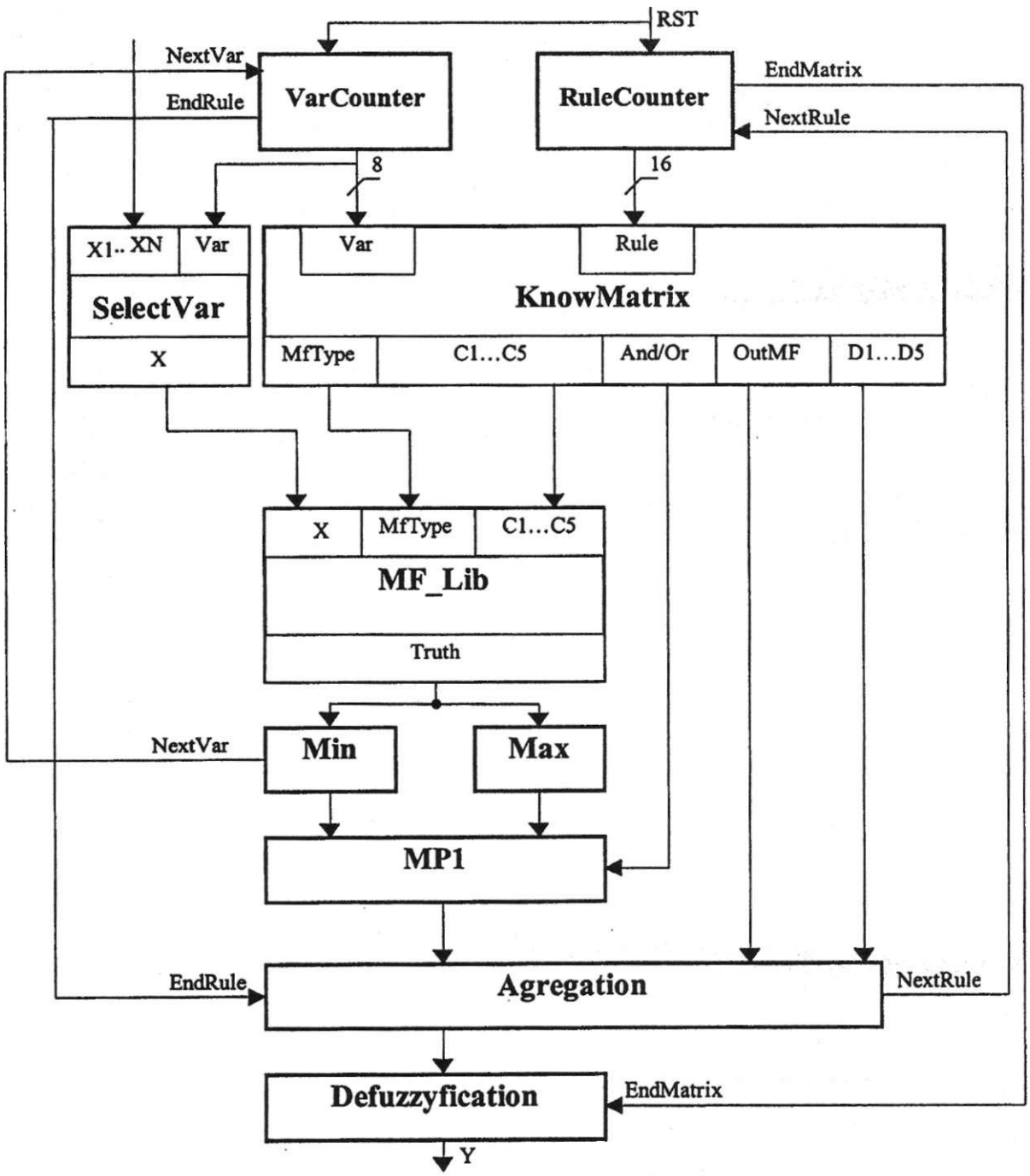



**Рис.** 8. Структура спеціалізованого фаззі-процесора з однією вихідною змінною

### *4.4. Алгоритм роботи й організаційна побудова блока MF_Lib*

Вхідними портами блока MFJLib є MFType (номер типу функції належності), Cl ***...CS*** (змінні коефіцієнти), X (значення незалежної змінної) та CLK (сигнал синхронізації). MFJLib включає в себе ряд блоків, що реалізують різноманітні форми функцій належності (VHDL-код для трикутної форми, блок trimf, наведено на рис. 9), а також мультиплексор, який залежно від сигналу MFType перемикає на вихід один з результатів.

Оскільки при розрахунках значень функцій належності різних форм використовується різне число тактів, то для досягнення синхронності роботи процесора рекомендується застосовувати в конкретному НПНР функції належності однакової форми для будь-яких координат. Цього обмеження можна уникнути шляхом розробки блоків обчислення функцій належності за допомогою розподіленої арифметики, що є самостійною задачею подальших досліджень.

### *4.5. Реалізація блока KnowMatrix*

Матриця знань та інформація про вхідні і вихідну координати НПНР зберігаються в блоці KnowMatrix у формі одновимірних масивів цілих чисел. Нижче наведено фрагмент VHDL-програми, що реалізує НПНР з 2 вхідними змінними, перша з яких визначається на шкалі з двох, а друга — з трьох лінгвістичних термів. Вихідна змінна визначається на шкалі з п'яти лінгвістичних термів. Всі функції належності мають трикутну форму.

```
Index: array (0 to 2) of integer range 0 to
65535 := (0, 2, 5);
Terms: array (0 to Q) of integer range 0 to 255 :=
(0, 0, 0, 255, 0, 0, 0, 0, 128, 255, 0, 0 ...);
Matrix: array (0 to W) of integer range 0 to 255 :=
(0, 0, 0, 0, 1, 0, 1, 0, ...);
```

Масив Terms зберігає інформацію про координати НПНР. Для кодування кожного лінгвістичного терму виділяється 6 байт. Перший байт вказує тип функції належності, решта 5 визначає значення змінних коефіцієнтів. У наведеному прикладі напівжирним шрифтом виділено визначення лінгвістичного терму з функцією належності трикутної форми (0 в першому байті) та значеннями коефіцієнтів 0, 0, 255.

Масив Index індексує масив Terms. Номер байта, з якого починається описування термів для і-ої змінної, визначається як Index(i)-6. Описування вихідної змінної починається з байта номер Index(n+1) ˙ 6

Матриця знань зберігається в масиві Matrix. Для кожного правила виділяється ***n + 2*** байт. Напівжирним на лістингу виділено описування правила, в якому перші ***л*** байт визначають ліву частину правила, байт номер ***n*** + 1 визначає праву частину правила, а байт номер ***n + 2*** визначає тип об'єднання логічних виразів у лівій частині правила ("0" відповідає AND, "1" — OR).

## 5. Тестування моделі поведінки НПНР

Для перевірки правильності роботи проведено моделювання розробленого проекту НПНР на функціональному рівні. В таблиці 3 наведено дані про координати тестового НПНР.

*Таблиця 3.* **Координати тестового НПНР**

| Назва | Тип | Кількість термів | Функції належності | |
|---|---|---|---|---|
| | | | Коефіцієнти | *Форма* |
| x1 | Вхідна | 3 | [0 0 255]<br>[0 128 255]<br>[0 255 255] | трикутні |
| x2 | Вхідна | 3 | [0 0 255]<br>[0 128 255]<br>[0 255 255] | трикутні |
| y | Вихідна | 3 | [0 0 255]<br>[0 128 255]<br>[0 255 255] | трикутні |

```
entity TriMF is
     port(
          X : in   STD_LOGIC_VECTOR(7 downto 0);
          C1: in   STD_LOGIC_VECTOR(7 downto 0);
         C2:  in   STD_LOGIC_VECTOR(7 downto 0);
         C3:  in   STD_LOGIC_VECTOR(7 downto 0);
         C4:  in   STD_LOGIC_VECTOR(7 downto 0);
         C5:  in   STD_LOGIC_VECTOR(7 downto 0);
         CLK: in   STD_LOGIC;
         Y : out STD_LOGIC_VECTOR(7 downto 0)
         );

end TriMF;

architecture MF21 of Mf21 is
begin
     process (CLK)
     variable a, res: integer range 0 to 255;
     variable Xmin: integer range 0 to 255;
     variable Xhigh: integer range 0 to 255;
     variable Xmax: integer range 0 to 255;
     Variable r1: STD_LOGIC_VECTOR(7 downto 0);
     begin
          a := CONV_INTEGER(X(7 DOWNTO 0));
          Xmin := CONV_INTEGER(C1(7 DOWNTO 0));
          Xhigh:= CONV_INTEGER(C2(7 DOWNTO 0));
          Xmax := CONV_INTEGER(C3(7 DOWNTO 0));
          if a<Xmin then
               res:=0;
               else if a<Xhigh then
                      res:=(255 * (a-Xmin))/(Xhigh-Xmin);
                    else if a<Xmax then
                      res:=(255*(a-Xmax))/(Xhigh-Xmax);
                    else
                     res:=0;
                   end if;
               end if;
          end if;
          Y<= CONV_STD_LOGIC_VECTOR(res, 8);
          end process;
end TriMF;
```

**Рис. 9.** VH DL-КОд для реалізації трикутних функцій належності

Результати моделювання наведені у вигляді характеристичної поверхні НППР на Рис. 10.

Слід відмітити повну зіставлюваність одержаних результатів з результатами попереднього дослідження еталонної моделі відповідного НППР.

Результати наукових досліджень свідчать, що використання програмованих логічних ІМС як елементної бази для проектування НППР та мов програмного опису апаратного забезпечення, зокрема VHDL, дозволить значно розширити сферу застосування систем підтримки прийняття рішень, синтезованих на основі універсальних НППР.

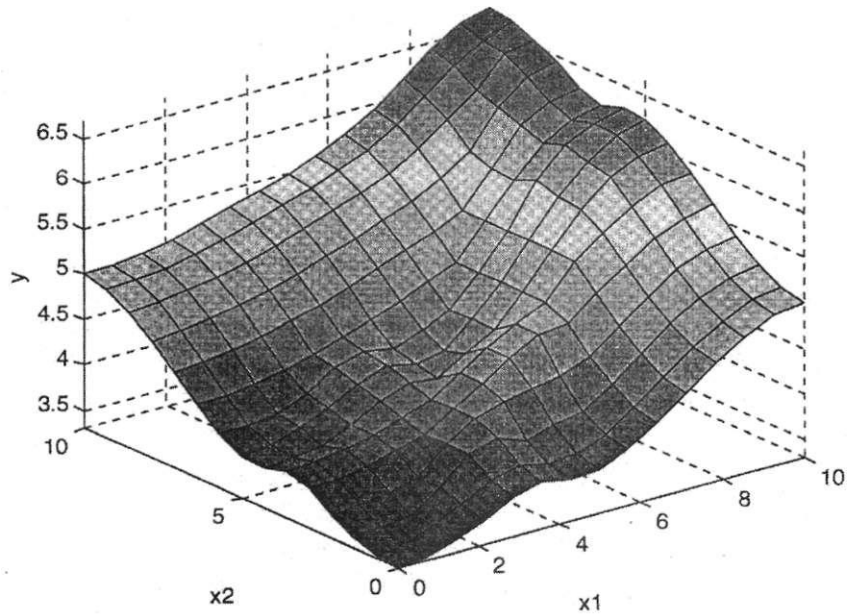



**Рис. 10.** Результата моделювання поведінки НППР

1. *Архангельский В. И., Богаенко И. Н., Грабовский Г. Г., Рюмшин Н. А.* Системы фуцци-управления.— К.: Техника, 1997,— 208 с.

2. *Кондратенко Ю. П., Сидоренко С. А.* Програмно-алгоритмічне забезпечення комп'ютеризованих систем технічної діагностики і прогнозування поведінки складних технічних об'єктів // Четверта всеукр. міжн. конф. "Обробленя сигналів і зображень та розпізнавання образів" (19—23 жовтня, Київ, Україна).— Київ, 1998.— С. 125—126.

3. *Кондратенко Ю. П., Сидоренко С. А.* Системи підтримки прийняття рішень на основі пристроїв з нечіткою логікою // Збірник наукових праць УДМТУ.— Миколаїв, УДМТУ, 1999.—Вип. 4.

4. *Кофман А., Хил-Алуха Х.* Введение теории нечетких множеств в управлении предприятиями.— Минск: Вышэйшая школа, 1992.— 224 с.

5. Программируемые логические ИМС на КМОП-структурах и их применение / П. П. Мальцев, Н. И. Гарбузов, А. П. Шарапов, Д. А. Кнышев.— М.:Энергоатомиздат, 1998.— 160 с.

6. *Ротштейн А. П.* Интеллектуальные технологии идентификации: нечеткие множества, генетические алгоритмы, нейронные сети.— Винница: "УНІВЕРСУМ-Вінниця", 1999.— 302 с.

7. *Циделко В., Хандоняк В.* Информационные технологии на базе нечеткой логики (Fuzzy logic) //Электронные компоненты и системы.— 1999.— № 10 (26).— С. 29—34.

8. Active-HDL User's Guide. Second edition.— Copyright ALDEC, Inc., 1999.— 213 p.

9. *Kondratenko Y. P., Sidorenko S. A.* Fuzzy models for forecasting of vessel's stability in extreme conditions // Thesis of Conf. Reports of International Conference "Dynamical Systems Modeling and Stability Investigation" (May 25—29, Kyiv, Ukraine).— Kyiv, 1999,— P. 84.

10. Test Access Port And Boundary-Scan Architecture // IEEE Standard 1149.1 — 1990 (Includes IEEE Standard 1149.1a— 1993).

11. *Zimmerman H. J.* Fuzzy Set Theory and Its Applications— Kluwer, Dordrecht, 1991.— 315 p.

*Kondratenko Y. P., Sidorenko S. A.*

# DESIGN METHODS OF THE FUZZY DECISION MAKING UNITS BASED ON THE PROGRAMMABLE LOGIC DEVICES PLD

The methods of the fuzzy decision making units designing and the possibilities of programmable logic devices (PLD) utilisation as a hardware base for elaboration of special-purpose fuzzy processors are considered in the present paper. Also the comparison analysis of FPGA and CPLD technologies is presented here. The approaches based on the using of ABEL, Verilog and VHDL programming languages are described in details. Furthermore authors offer the method of the multipurpose FDMD designing by means of Active-HDL 3.5 software. The structure of single FDMD units and their VHDL programme implementations are presented also. The results of FDMD behaviour simulation confirm the efficiency of PLD using for automatisation of decision-making processes in uncertain conditions.