Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Алгоритм функціонування вдосконаленого керуючого автомату паралельної дії

(тема)

Виконав:

студент II курсу, групи СПм-22-3

Коломоєць В. С.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Бовчалюк С. Я.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Коломойцю Владиславу Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Алгоритм функціонування вдосконаленого керуючого автомату паралельної дії

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи 1) інформаційна технологія паралельного логічного керування
2) Структура класичного логічного керуючого автомату паралельної дії
3) Структура ЛКА ПД із розширеними функціональними можливостями
4) Технічна документація щодо практичної реалізації керуючих структур з паралельною архітектурою

4. Перелік питань, що потрібно опрацювати у роботі
1) Аналіз структурної організації керуючих автоматів з паралельною архітектурою
2) Дослідження математичної моделі та структури ЛКА ПД
3) Аналіз структури керуючих автоматів із функціями вбудованих програмованих таймерів
4) Дослідження універсального алгоритму функціонування керуючого автомата паралельної дії
5) Розробка алгоритму функціонування перспективного ЛКА ПД
6) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз структурної організації керуючих	02.04.24-08.04.24	
	автоматів з паралельною архітектурою		
2	Дослідження математичної моделі та	09.04.24-16.04.24	
	структури ЛКА ПД		
3	Аналіз структури керуючих автоматів із	17.04.24-27.04.24	
	функціями вбудованих таймерів		
4	Дослідження універсального алгоритму	28.04.24-06.05.24	
	функціонування ЛКА ПД		
5	Розробка алгоритму функціонування	07.05.24-27.05.24	
	перспективного ЛКА ПД		
6	Оформлення матеріалів кваліфікаційної роботи	28.05.24-05.06.24	
7	Подання кваліфікаційної роботи керівникові	06.06.24-08.06.24	
	та її попередній захист		
8	Подання кваліфікаційної роботи на	09.06.24-12.06.24	
	рецензування		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Бовчалюк С. Я.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 73 с., 13 рис., 1 табл., 2 дод., 24 джерела.

АЛГОРИТМ, КЕРУЮЧІ ПРИСТРОЇ З ПАРАЛЕЛЬНОЮ АРХІТЕКТУРОЮ, ЛОГІЧНИЙ КЕРУЮЧИЙ АВТОМАТ ПАРАЛЕЛЬНОЇ ДІЇ, ПЛІС-КОНТРОЛЕР ПАРАЛЕЛЬНОЇ ДІЇ, СТРУКТУРА ПЛК ПД, HDL-МОДЕЛЬ

Метою кваліфікаційної роботи є розробка алгоритму функціонування керуючого автомату паралельної дії з розширеним функціоналом у вигляді програмованих таймерів.

У ході виконання кваліфікаційної роботи проведено аналіз структурної організації керуючих автоматів з паралельною архітектурою, їх математичної моделі та структури.

Виконано дослідження особливостей побудови ЛКА ПД із функціями вбудованих програмованих таймерів. Проведено аналіз універсального алгоритму функціонування класичного логічного керуючого автомату паралельної дії та шляхи його вдосконалення.

Розроблено алгоритм функціонування перспективного ЛКА ПД з розширеним функціоналом у вигляді програмованих користувачем таймерів.

ABSTRACT

Master’s thesis: 73 pages, 13 figures, 1 tables, 2 appendices, 24 sources.

ALGORITHM, CONTROL DEVICES WITH PARALLEL ARCHITECTURE, LOGIC CONTROLLER OF PARALLEL ACTION, CPLD-CONTROLLER OF PARALLEL ACTION, STRUCTURE OF PLC PA, HDL-MODEL

The purpose of the qualification work is to develop an algorithm for the operation of a parallel-action control automaton with extended functionality in the form of programmable timers.

In the course of the qualification work, an analysis of the structural organization of control automata with parallel architecture, their mathematical model and structure was carried out.

The study of the peculiarities of the construction of the PLC PA with the functions of built-in programmable timers was carried out. The analysis of the universal algorithm of the functioning of the classical logical control automaton of parallel action and the ways of its improvement was carried out.

An algorithm for the operation of a promising PLC PA with extended functionality in the form of user-programmable timers has been developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БВО – блок вибору операції

БІ – блок індикації

БЛК – блок логічного керування

БП – блок пам’яті

БПВЗ – блок пам'яті внутрішніх змінних

БПЗК – блок пам'яті заборонених комбінацій

БПК – блок пам’яті команд

БПП – блок пам’яті переходів

БПС – блок пам’яті станів

ВІС – велика інтегральна схема

ВЛК – вузол логічного керування

ВР – вихідний регістр

ІМС – інтегральна мікросхема

ЛА – лічильник адреси

ЛКА – логічний керуючий автомат

ПД – паралельної дії

ПЛІС – програмована логічна інтегральна схема

ПЛК – програмований логічний контролер

ПЛК ПД – програмований логічний контролер паралельної дії

ППЛК – паралельний програмований логічний контролер

САПР – система автоматизованого проектування

СП – схема порівняння

ТА – технологічний агрегат

$ТА_{ДЦ}$ – технологічний агрегат дискретної циклічної дії

AHDL – мова програмування апаратури компанії Altera (англ., Altera Hardware Description Language)

CPLD – програмована логічна інтегральна схема (англ., Complex Programmable Logic Device)

FPGA – програмована користувачем вентильна матриця (англ., Field-Programmable Gate Array)

HDL – мова програмування апаратури (англ., Hardware Description Language)

TVP – технологічне візуальне програмування (англ., Technological Visual Programming)

ВСТУП

Ні у кого не викликає сумніву, що автоматизація технологічних процесів є одним із найпотужніших факторів підвищення ефективності роботи промислових підприємств. Водночас темпи впровадження сучасних систем керування на базі мікроелектронних та мікропроцесорних пристроїв у деяких галузях залишаються невисокими. Насамперед це характерно для систем керування відповідальними технологічними процесами, де відмова (як апаратна, так і програмна) може спричинити загибель людей або значні економічні втрати.

У роботах [3-5, 20] показано можливості щодо суттєвого підвищення якості розробки і функціонування керуючих та обчислювальних систем і пристроїв, що може бути досягнуто на основі використання у якості елементної бази регулярних мікроелектронних структур, а саме – програмованих логічних інтегральних схем (ПЛІС) і ПЛІС-контролерів, що побудовані на їх базі. Останніми роками питанням дослідження і розвитку інформаційної технології паралельного логічного керування на базі ПЛІС-контролерів присвячувалось недостатньо уваги. Але все ж таки у тих нечастих наукових роботах, де ці питання розглядались, було показано, що розвиток цієї технології має достатньо широкі перспективи [3, 6, 24]. Уведення до ПЛІС-контролерів паралельної дії, як ядра такої технології, додаткових функцій, може дозволити у значній мірі розширити сфери її застосування. За рахунок цього з’являється можливість її більш широкого застосування не тільки для керування відповідальними об’єктами, або об’єктами критичної дії, але і для побудови звичайних систем керування промисловими або індустріальними об’єктами і технологічними лініями.

Аналіз публікацій , що стосуються інформаційної технологій керування паралельної дії взагалі та ПЛІС-контролерів зокрема показав, що існує три основних напрямки для розвитку керуючих пристроїв розглядаємого класу на

базі ПЛІС. Ось ці три основних напрямки:

- реалізація програмованих логічних контролерів паралельної дії на елементній базі більшої ступені інтеграції – від дискретних ІМС до надвеликих ВІС, а саме сучасних кристалів програмованих логічних інтегральних схем найновіших серій [2, 5, 8, 18];

- розвиток і розширення функціоналу внутрішньої структури керуючих структур паралельної дії. Серед прикладів такого розвитку можна вказати розширення безпекових спроможностей керуючих систем, що побудовані на їх базі. Такі спроможності з’явились за рахунок уведення функцій, що забезпечують унеможливлення видачі на керований об’єкт заборонених комбінацій команд керування; розширення інструментарію логічної обробки потоку вхідних сигналів, через додавання можливості їх аналізу не тільки у диз’юнктивній формі, але і у кон’юнктивній; уведення елементів що дозволяють реалізувати функції нечіткого логічного висновку [2, 3, 5, 6, 12];

- вдосконалення мови і технології програмування керуючих пристроїв з паралельною архітектурою. Справа в тому, що стиль роботи більшості сучасних програмістів не відповідає рівню вимог до програмних продуктів. Програмісти схильні сприймати свій вид діяльності як мистецтво, спираючись насамперед на власну інтуїцію, своє особисте, суб'єктивне сприйняття вирішуваних ними завдань. Такий підхід (який на сьогодні цілком можна вважати загальноприйнятим) часто призводить до появи помилок у програмному забезпеченні, що, своєю чергою, стає причиною важких наслідків. У зв'язку з цим, набуває все більшої актуальності певною мірою парадоксального питання: чи взагалі можна виключити програміста з процесу підготовки ПЗ або, принаймні, звести його участь і вплив на результат до мінімуму. Прикладами реалізації такого підходу є поступова трансформація технології формування керуючої програми для ППЛК від її формування на паперовому носії мовою ЯППЛК з подальшим ручним програмуванням ІМС ПЗП, до більш розвиненої мови ЯПЛК-М, що дозволяла формувати керуючу програму із застосуванням ЕОМ [6, 18].

Також у цьому контексті слід згадати створення і кристалізацію автоматизованої технології програмування TVP (Technological Visual Programming) різних версій, від найбільш простої, орієнтованої на автоматизоване формування керуючої програми для класичних контролерів послідовної дії – TVP 1.0, до її розширеної версії TVP 2.0, що дозволяє формувати непідготовленому (з точки зору знань технології класичного програмування керуючих пристроїв) користувачеві керуючу програму для ПЛІС-контролера ПД мовою ЯПЛК-М [1, 23].

Але все ж таки одним із найбільш перспективних вдосконалень, що дозволяє значно розширити функціонал керуючих пристроїв з паралельною архітектурою на базі ПЛІС є уведення до їх структури внутрішніх програмованих користувачем таймерів і лічильників. В одній з останніх публікацій за цим науковим напрямком, було показано, що саме уведення програмованих таймерів є актуальною і пріоритетною задачею на даному етапі. У колективній науковій публікації [9] було запропоновано вдосконалену архітектуру ЛКА ПД, що містить у своєму складі елементи, що реалізують функціонал програмованих таймерів. У той самий час, підтримки з боку алгоритму функціонування надано не було.

Таким чином розробка алгоритму функціонування керуючих автоматів паралельної дії з розширеними функціональними можливостями у вигляді програмованих користувачем таймерів, є, на даному етапі досліджень, пріоритетною і актуальною задачею.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Структурна організація керуючих автоматів з паралельною архітектурою

У наукових публікаціях [5, 6, 20] було запропоновано і детально розглянуто математичну модель та архітектуру логічних керуючих автоматів і пристроїв з паралельним принципом обробки потоку вхідних даних і паралельним же принципом формування команд керування виконавчими механізмами технологічного агрегату. На базі цих моделі і архітектури було розроблено, реалізовано, впроваджено у промислове виробництво функціонально завершені програмовані логічні контролери паралельної дії (ПЛК ПД). У літературі цей клас керуючих пристроїв отримав загальновживану назву – паралельні програмовані логічні контролери або ППЛК, характерною особливістю яких було те, що вони були реалізовані на ІМС малої степені інтеграції, тобто фактично на дискретних елементах. Аналіз та досвід практичного застосування таких структур показав, що вони дійсно характеризуються безумовними перевагами перед класичними ПЛК послідовної дії, як у швидкодії, так і у надійності функціонування, але, у той самий час, мають дуже суттєві недоліки. Першим недоліком можна вважати обмежені можливості логічного аналізу потоку станів входів. Цей недолік може призводити до необґрунтованого збільшення часу, що витрачається користувачем обладнання на формування керуючої програми. Другим недоліком є відсутність можливості виявлення та заборони видачі аварійних комбінацій вихідних сигналів. Цей недолік є критичним у випадку реалізації системи керування відповідальним технологічним обладнанням, виникнення аварій на якому є критичним. Третім недоліком є відсутність автоматичного запису програми до пам’яті керуючого пристрою. Тобто формування

керуючої програми відбувалось, фактично, у суто ручному режимі, тобто без застосування ЕОМ. На базі вказаного матеріалу було розроблено математичну модель, та побудовано більш досконалі архітектури засобів керування та формування інформаційної технології паралельного логічного керування,в яких враховано розглянуті недоліки базових моделей, архітектур та технології. Ці моделі та архітектури були в основному орієнтовані на створення систем керування об’єктами критичного застосування, але, у той самий час, могли бути ефективно використані і для реалізації таких систем у звичайному індустріальному секторі.

1.1.1 Математична модель ЛКА ПД

В основу методології побудови керуючих автоматів із паралельним принципом обробки входів і формування виходів, покладено раціональне поєднання та практичне використання властивостей регулярності. Вказана властивість притаманна багатьом об’єктам у різних сферах людської діяльності, але з точки зору побудови систем керування, важливим є притаманність основним елементам, з яких вона може бути побудована. А саме: об’єкти керування дискретної циклічної дії ($ТА_{ДЦ}$); технологічні мови опису алгоритмів керування $ТА_{ДЦ}$; матричні мікроелектронні структури, або за сучасною класифікацією – програмовані логічні інтегральні схеми (ПЛІС) [5, 20].

Для адекватного опису алгоритму функціонування ЛКА ПД, розглянемо їх математичну модель, основні принципи функціонування і архітектуру.

Розгляд необхідно розпочати з тієї різниці у базових принципах функціонування, що формує можливість чіткої диференціації між керуючими автоматами паралельної та послідовної дії.

Характеристикою, що формує принципову відмінність між ЛКА послідовної і паралельної дії можна вважати тривалість циклу однократного

обслуговування входів, що контролюються автоматом і виходів керування виконавчими механізмами технологічного агрегату:

$$T_ц = t_I + t_{II}, \tag{1.1}$$

де $T_ц$ – сумарний параметр, що визначає тривалість циклу однократного обслуговування всіх входів та всіх виходів автомата в одиницях дискретного автоматного часу;

t_I – значення дискретного автоматного часу, що витрачається ним на аналіз станів вхідних сигналів;

t_{II} – значення дискретного автоматного часу, що витрачається автоматом на формування команд керування виконавчими механізмами технологічного агрегату.

Введемо поняття логічних контролерів послідовної і паралельної дії. ПЛК послідовної дії – це контролери в яких використовується послідовний принцип обслуговування контрольованих входів і керованих виходів. Дискретний автоматний час для таких контролерів можна визначити наступним виразом:

$$t^{I} = \sum_{i=1}^{k} t_i^{I}, \qquad t^{II} = \sum_{j=1}^{m} t_j^{II}, \tag{1.2}$$

де k – загальна кількість контрольованих входів контролера;

m – загальна кількість команд керування виконавчими механізмами технологічного агрегату.

На відміну від послідовних ПЛК, паралельними контролерами будемо називати такі, у яких використовується паралельний принцип обслуговування входів-виходів. Для таких контролерів дискретний

автоматний час обслуговування контрольованих входів і обслуговуємих виходів визначається виразами (1.3):

$$t^{I}=t_i^{I}(i=1, 2, \dots, k), \qquad t^{II}=t_j^{II}(j=1, 2, \dots, m). \tag{1.3}$$

Таким чином можна чітко побачити, що при використанні паралельного принципу побудови ПЛК час на обслуговування входів-виходів не залежить від їх кількості.

Перш ніж переходити безпосередньо до розгляду математичної моделі. необхідно чітко визначити поняття технологічного агрегату дискретної циклічної дії ($ТА_{ДЦ}$), що вже неодноразово згадувалось раніше.

Технологічними агрегатами дискретної циклічної дії будемо називати такі об’єкти, поведінка яких у часі та просторі є строго детермінованою і може бути описана двома кінцевими множинами станів: множиною станів, у яких протягом циклу роботи агрегату знаходяться механізми і множиною станів датчиків, що контролюють стани відповідних механізмів.

При цьому множина станів, у яких протягом циклу знаходяться механізми визначається як (1.4), а множина що визначає контроль відповідних механізмівяк, як (1.5).

$$C=\{c_1, c_2, \dots, c_m\} \tag{1.4}$$

$$A=\{a_1, a_2, \dots, a_k\}. \tag{1.5}$$

З урахуванням вищевказаного, цикл роботи $ТА_{ДЦ}$ можемо визначити кінцевою кількістю інтервалів дискретного автоматного часу (2.6)

$$T=\{t_1, t_2, \dots, t_s\}, \tag{1.6}$$

Окремо слід зауважити, що для кожного *i*-го інтервалу існує єдина кінцева підмножина (іншими словами комбінація) станів датчиків (1.7), що є єдиною, (іншими словами «дозволеною») для вмикання відповідної кінцевої підмножини (або комбінації) механізмів (1.8)

$$A_i=\{a_{i1}, a_{i2}, \dots , a_{ik}\}, A_i \subset A, \tag{1.7}$$

$$C_i=\{c_{i1}, c_{i2}, \dots ,c_{im}\}\ C_i \subset C. \tag{1.8}$$

Таким чином циклограму роботи $ТА_{ДЦ}$ можна визначити або задати двома прямокутними матрицями, що мають кінцеві розміри. Перша матриця – матриця станів датчиків $|M_A|$ (1.9). Друга матриця – і матриця станів механізмів $|M_C|$ (1.10). Окремо зазначимо, що вектори-рядки у цих матрицях розміщені строго детерміновано, та кожному *i*-му рядку матриці $|M_A|$ однозначно відповідає *i*-й рядок матриці $|M_C|$:

$$|M_A|=\begin{Vmatrix} a_{11}\ a_{12}\ \dots a_{1k} \\ a_{21}\ a_{22}\ \dots a_{2k} \\ \dots\ \dots\ \dots\ \dots \\ a_{s1}\ a_{s2}\ \dots a_{sk} \end{Vmatrix}; \tag{1.9}$$

$$|M_C|=\begin{Vmatrix} c_{11}\ c_{12}\ \dots c_{1m} \\ c_{21}\ c_{22}\ \dots c_{2m} \\ \dots\ \dots\ \dots\ \dots \\ c_{s1}\ c_{s2}\ \dots c_{sm} \end{Vmatrix}. \tag{1.10}$$

Кількість датчиків керуючого автомату, що встановлені на механізмах технологічного агрегату, визначає кількість стовпців *k* матриці $|M_A|$. У той час, як кількість виконавчих механізмів, що контролюються цими датчиками.

однозначно визначає кількість стовпців m у матриці $|M_C|$. Кількість рядків, або етапів циклограми. або іншими словами – кількість рядків, що визначає кроки програмного керування, однозначно задає кількість рядків s в обох матрицях. При цьому кількість рядків в обох матрицях є однаковою.

Окремо зазначимо, що буде розглядатись лише ситуація, коли стани датчиків і виконавчих механізмів $ТА_{ДЦ}$ можуть приймати лише два кінцевих фіксованих значення – «включено» або «виключено». Така ситуація може бути адекватна описана категоріями булевої алгебри, отже і матриці $|M_A|$ та $|M_C|$ розглядаємої математичної моделі є булевими, з детермінованим розміщенням у них векторів-рядків станів механізмів і векторів-рядків станів відповідних датчиків.

Окрім матриць, для опису математичної моделі, введемо вектори, взаємодія яких дозволить надати формальний опис функціонування паралельних логічних керуючих автоматів.

Вектор a – або вектор станів детермінованих входів. Цей вектор призначений для фіксації комбінації станів датчиків або органів керування $ТА_{ДЦ}$, значення яких можна чітко і однозначно визначити у часі, тобто вони є детермінованими. Довжину цього вектора позначимо змінною k.

Вектор $в$ – вектор умов (вектор станів стохастичних входів) Це комбінація станів зовнішнього середовища тобто станів датчиків або органів керування, яка не є заданою заздалегідь. Тобто ця комбінація є стохастичною. Зазвичай вектор $в$ визначає умови розгалуження алгоритму керування обладнанням. Позначимо довжину цього вектора – u.

Вектор c – вектор керування виконавчими механізмами. Вектор, що визначається комбінацією сигналів, що має бути видана на виконавчі механізми $ТА_{ДЦ}$. Позначимо довжину цього вектора літерою – m.

Ще два вектори визначають внутрішні стани керуючого автомату і призначені для формування адреси переходу і забезпечення додаткових безпекових функцій ЛКА ПД.

Вектор d – вектор задавання адреси переходу. Вектор, що визначає

початкову адресу підпрограми, якщо вона має розгалуження. Фактично цей вектор являє собою номер рядка матриць $|M_A|$ та $|M_C|$ станів датчиків і станів механізмів. Довжину вектора d позначимо літерою q.

Останній вектор – вектор e, або вектор заборонених станів команд керування. Цей вектор було уведено до складу математичної моделі удосконаленого ЛКА ПД для забезпечення блокування заборонених комбінацій команд керування виконавчими механізмами $ТА_{ДЦ}$, поява яких на виході ЛКА однозначно призведе до аварійного ситуації на керованому об'єкті. Довжина цього вектора співпадає з довжиною вектора c команд керування, і позначається m.

Розглянемо процедуру формування матриць математичної моделі. Множина векторів a_i формує матрицю $|M_A|$ – матрицю станів. Розмір цієї матриці – $s \times к$.

Матрицю команд $|M_C|$ сформовано множиною векторів c_i. Розмір матриці – $s \times m$.

Матриці станів і команд виконують функцію збереження очікуваних станів датчиків і команд керування ТА. Важливим є те, що у кожний конкретний момент дискретного автоматного часу конкретному рядку матриць $|M_A|$ та $|M_C|$ ставиться у відповідність певний вектор a і c.

Розглянемо наступну матрицю моделі – матрицю $|M_B|$ адрес переходів. Розмір матриці $q \times u$ і вона складається з рядків, кожний з яких є умовою переходу до підпрограми. У загальному випадку, якщо алгоритм функціонування ТА не передбачає розгалужень. тобто містить лише одну лінійну програму, матриця $|M_B|$ може бути відсутня. У випадку наявності підпрограм, кількість рядків матриці B визначається їхньою кількістю. Кількість стовпців матриці $|M_B|$ однозначно відповідає довжині вектора $в$. Зазначимо, що у процесі роботи ЛКА фактично відбувається відображення рядка матриці $|M_B|$ у вектор d. У загальному випадку довжини векторів $в$ і d у не обов'язково мають співпадати, тобто $u \neq log_2(s)$.

Ще однією матрицею, що присутня у моделі удосконаленого

керуючого автомату, є матриця $|M_E|$ – заборонених станів. Такий автомат характеризується покращеними показниками надійності роботи керованого об’єкта. Така його особливість визначається тим, що автоматом унеможливлюється видача керуючих комбінації вихідних сигналів, які можуть призвести до аварійного стану технологічного агрегату. Число рядків і стовпців матриці визначається тим набором станів і команд керування технологічного об’єкта, що можуть призвести до аварії. В загальному випадку розмір матриці $|M_E|$ дорівнює $r \times m$, де m – кількість керуючих сигналів, а r – кількість заборонених комбінацій керуючих сигналів.

Розглянемо принцип функціонування логічного керуючого автомату паралельної дії, який може бути визначений взаємодією розглянутих раніше векторів.

Роботу автомату варто розпочати з припущення, що він знаходиться на певному абстрактному p-му кроці алгоритму функціонування технологічного об’єкта у проміжку між моментами часу між t_p і t_{p+1}. У цей момент вектор керувань $C(t_p)$ формує значення компонент $c_j(t_p)$, які зчитано з i-го рядка матриці програми керування C

Припустимо, що на p-му кроці алгоритму функціонування керованого об’єкта у проміжку між моментами часу t_p і t_{p+1} вектор керувань $C(t_p)$ отримує значення компонент $c_j(t_p)$, що зчитані з i-го рядка матриці програми керування $|M_C|$. Вектор керувань $C(t_p)$ має зберігати значення цих компонент до початку наступного, тобто $p+1$-го кроку алгоритму (1.11)

$$c_j(t_p) = C_{ij}, \tag{1.11}$$

У даному виразі індекс j приймає значення від 1 до m. Номер рядка визначається на $(p-1)$-ому кроці алгоритму:

$$I = g(p\text{-}1), \tag{1.12}$$

де $g(p)$ – функція кроку алгоритму p.

Першим етапом після формування вектора команд керування, є його перевірка на наявність забороненої комбінації вихідних сигналів. Для цього значення вектора $c_j(t_p)$ мають бути поелементно перевірені на співпадіння зі значенням векторів e_{xj} $(x=1, 2, ..., r; j=1, 2, ..., m)$, які зберігаються у матриці $|M_E|$. Така перевірка формує вираз (2.13).

$$\varepsilon_p = \bigvee_{x=1}^{r} [\bigwedge_{j=1}^{m} (c_{ij} = e_{xj})] \,. \tag{1.13}$$

На цьому етапі можливі дві ситуації: булева функція ε приймає значення логічної одиниці або нуля. У випадку $\varepsilon=1$ відбувається процедура блокування видачі комбінації команд керування на технологічний агрегат, тобто блокування видачі вектора $c_j(t_p)$. Це пояснюється тим, що функція ε є ознакою наявності забороненої комбінації вихідних сигналів. На цьому етапі робота технологічного агрегату припиняється до тих пір, поки не буде усунуто умови, що породили появу забороненої комбінації.

У випадку, коли булева функція ε приймає значення логічного нуля, керуючі впливи мають призвести до зміни стану керованого об’єкта. Цей процес контролюється системою відповідних датчиків, інформацію від яких отримує ЛКА. У математичній моделі така зміна має відображується у зміні координат вектора станів $a(t_p)$.

Необхідно зазначити, що видача вектора $c_j(t_p)$на реальному об’єкті керування не обов’язково призводить до зміни вектора $a(t_p)$. Це може бути пов’язано з неполадками або збоями у роботи ТА. Отже необхідно виконувати безперервне порівняння станів детермінованих входів $a_j(t_p)$ $(j=1, 2, ..., k)$ з їх очікуваними значеннями A_{ij} $(j=1, 2, ... , k)$, які записані до i-го

рядку матриці станів $|M_A|$.

Сама процедура порівняння може відбуватись за логікою «І» або за логікою «АБО». Вибір логіки порівняння визначається значенням ознаки F_p. У випадку якщо перехід має відбуватись при співпадінні усіх фактичних станів датчиків циклу з їх очікуваними значеннями, спрацювання яких очікується на p-му кроці програми, значення ознаки $F_p=1$ і це вмикає логіку роботи «І». Якщо перехід до наступного кроку алгоритму має відбутись при наявності сигналу хоча б від одного з датчиків, значення ознаки $F_p=0$ і це вмикає логіку роботи «АБО».

Логіка означеного вищеозначеного порівняння визначається наступною функцією:

$$\alpha_p = \begin{cases} \bigwedge_{j=1}^{k} (a_j(t_p) = A_{ij}), \text{ при } F_p = 0 \\ \bigvee_{j=1}^{k} (a_j(t_p) = A_{ij}), \text{ при } F_p = 1 \end{cases}. \qquad (1.14)$$

Якщо функція α_p приймає значення нуля. то це означає, що стан керованого об’єкта не змінився під дією керуючих впливів і, як наслідок, номер рядка матриці програми не змінюється $I=g(p-1)$.

Якщо ж булева функція α_p приймає значення логічної одиниці, то відбувається перехід до наступного рядка поточної підпрограми у відповідності до виразу (1.15).

$$I = g(p) = g(p-1)+1, \text{ якщо } (ST_p \vee Int1_p) = 0, \qquad (1.15)$$

де ST_p – ознака дозволу переходу;

$Int1_p$ – ознака безумовного переходу.

Розглянемо призначення ознаки дозволу перходу ST_p. Алгоритми

можна розділити на три великих класи:

- нерозгалужені алгоритми;
- алгоритми із розгалуженнями;
- алгоритми із вкладеними циклами.

Дослідження всіх цих класів алгоритмів показує, що аналіз можливості переходу за адресою, яка не дорівнює попередньої адреси плюс одиниця, можна звести до перевірки наявності лише однієї ознаки яку у розглядаємій математичній моделі названо *ST* – дозвіл переходу.

Якщо ST_p приймає значення логічного нуля, це означає, що на поточному *p*-му кроці алгоритму виконується поточна підпрограма і необхідно виконати перехід до її наступного кроку. Якщо ST_p має значення логічної одиниці, то має відбутися перехід за адресою, що визначається станом стохастичних входів $b_j(t_p)$. тобто має відбутись перхід до виконання іншої підпрограми.

Окремо розглянемо призначення ознаки переривання $Int1_p$. ЇЇ наявність вказує на те, що на поточному кроці алгоритму виникла ситуація, при якій необхідно невідкладно почати відпрацювання іншої підпрограми. Це може підпрограма аварійної зупинки обладнання або підпрограма екстреного допрацювання циклу, тощо.

Якщо розглядати неспецифічні технологічні процеси, то процес відпрацювання алгоритму їх керування полягає, зазвичай, у послідовному виконанні рядків програми. тобто робота відбувається у відповідності до виразу $I=g(p-1)+1$. І лише у місцях розгалуження, номер рядка може збільшуватись не на одиницю, а приймає будь яке значення адреси переходу, які визначаються вектором на стохастичних входах.

Для формування такої адреси переходу у програмах з розгалуженнями відбувається порівняння станів стохастичних входів $b_j(t_p)$ з умовами B_{xj}, які зберігаються у матриці $|M_B|$ адрес переходів. У такому випадку відбувається формування булевої функції β_p:

$$\beta_p = \bigvee_{x=1}^{n} [\bigwedge_{j=1}^{u} (b_j(t_p) = B_{xj})] , \quad (1.16)$$

Індекси *j* і *x* виразу (2.16) пробігають значення від 1 до *u* і *n* відповідно. Отже виконується процедура одночасного аналізу усієї матриці $|M_B|$.

У ситуації, якщо булева функція (1.16) приймає значення логічної одиниці відбувається перехід до певного рядка нового *(p+1)*-го кроку алгоритму. що є першим рядком наступної підпрограми. Цей процес описується наступним виразом:

$$I = g(p) = [d_p = f(b_p)], \text{ якщо } (b_p \wedge ST_p) \vee Int1_p = 1, \quad (1.17)$$

де $d_p = f(b_p)$ – вектор адреси переходу.

Як і у випадку з виразом (1.15), потрібно обов'язково враховати значення ознак ST_p та $Int1_p$. Якщо ST_p має значення логічного нуля, то це означає, що на *p*-му кроці виконується поточна підпрограма, це блокує перехід до наступної підпрограми, до закінчення її відпрацювання. У такому випадку, згідно виразу (1.17) крок алгоритму визначається тільки станом детермінованих входів $a_j(t_p)$. Якщо на поточному *p*-му кроці алгоритму сформувалась ситуація, при якій необхідно невідкладно почати відпрацювання іншої. наприклад аварійної підпрограми, то значення переривання $Int1_p$ приймає значення логічної одиниці і перехід відбувається без урахування значення ознаки ST_p.

При переході до наступного *(p+1)*-го кроку алгоритму керуючі впливи і очікувані стани визначаються *g(p)*-м рядком матриць $|M_A|$ та $|M_C|$. У такому випадку цикл повторюється аналогічно до розглянутого вище алгоритму. але вже з іншими значеннями компонент векторів.

1.1.2 Структура класичного керуючого автомату паралельної дії

У контексті поточної роботи структура ЛКА ПД може бути представлена у декількох різних варіантах, але оскільки основною метою є розробка вдосконаленого алгоритму його роботи, то окрім класичного представлення архітектури, буде розглянуто і її варіацію у матричному вигляді

На рисунку 1.1 показано архітектуру ЛКА ПД у його класичному представленні [5] в. Саме така архітектура була закладена в основу промислових зразків ПЛІС-контролерів паралельної дії і реалізована на кристалах ПЛІС компанії Altera (рисунок 1.2) [8].

ЛКА паралельної дії складається з наступних елементів (рисунок 1.1).

Основу складає чотири блоки пам’яті:

- БПС – блок пам’яті станів контрольованих входів, що призначений для збереження очікуваних станів зовнішнього середовища;

- БПК – блок пам’яті команд керування обладнанням, що містить у собі набір команд керування технологічним агрегатом;

- БПП – блок пам’яті адрес переходів до підпрограм, який призначений для зберігання адрес переходів до підпрограм, що визначаються станом стохастичних входів;

- БПЗК блок пам’яті заборонених комбінацій команд керування, до якого записано комбінації команд керування, видача яких на технологічний агрегат є неприпустимою і має бути обов’язково заблокована.

Окрім блоків пам’яті ПЛК ПД включає в себе блоки, що формують загальну логіку роботи, функції індикації зв’язку з зовнішнім середовищем, тощо. Розглянемо їх, з вказівкою основних виконуваних функцій:

- БІ – блок індикації. Призначений для відображення інформації про неспівпадіння значень компонент вектора $a_j(t_p)$ $(j=1, 2, ..., k)$ з їх очікуваними значеннями A_{ij} $(j=1, 2, ... , k)$, які записані у блок пам’яті станів. Ця інформація може бути використана для діагностики обладнання

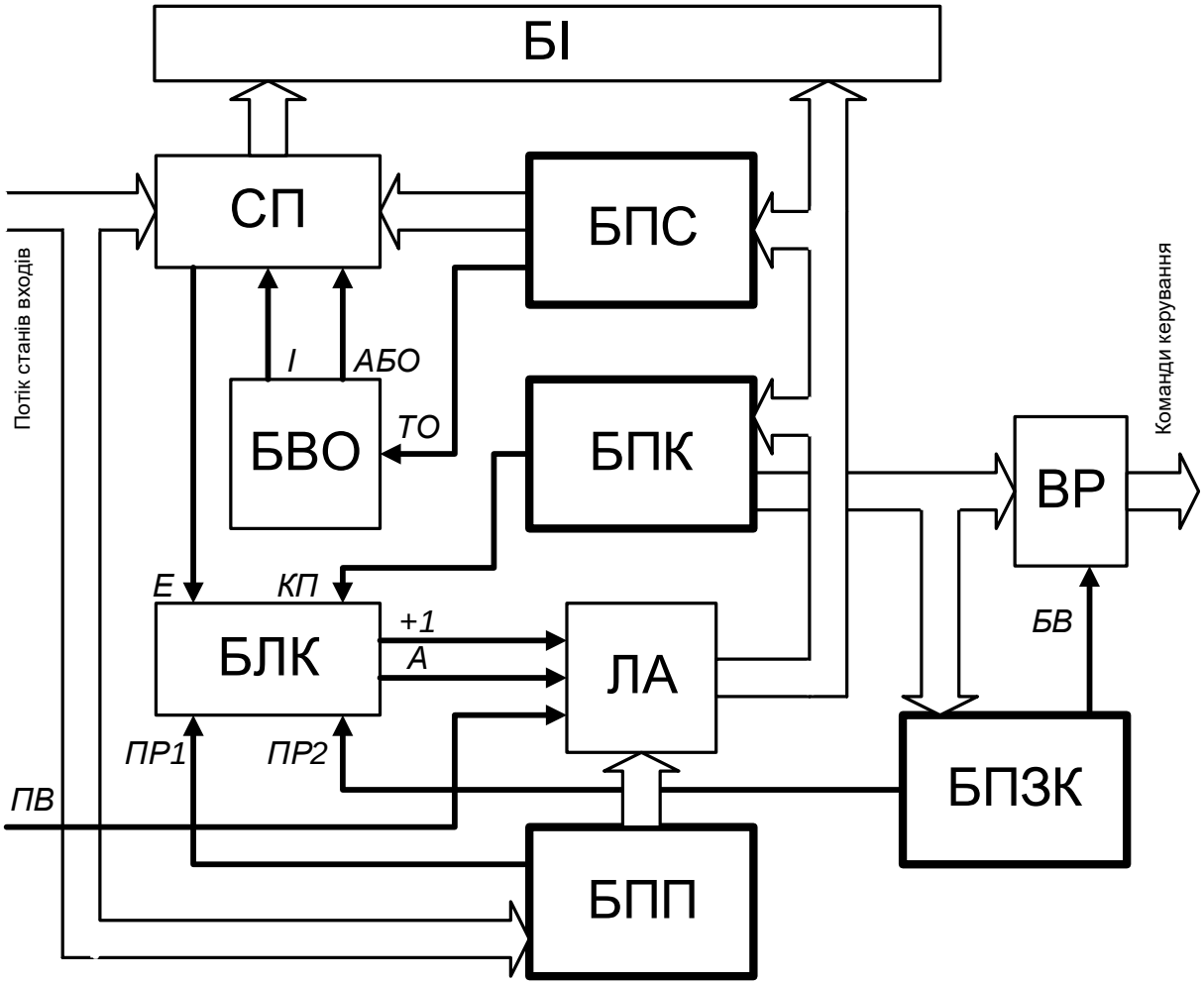


Рисунок 1.1 – Архітектура логічного керуючого автомату паралельної дії

- СП – схема порівняння. Виконує функції реалізації рівняння (1.14), тобто реалізує порівняння компонент вектора станів з їх очікуваними значеннями з БПС;

- БВО – блок вибору операції. Призначений для реалізації ознаки F_p, що входить до складу виразу (1.14) математичної моделі ЛКА ПД. Блок перемикає схему порівняння на роботу за логікою «І», або за логікою «АБО»;

- БЛК – блок логічного керування. Формує загальну логіку роботи ЛКА ПД і взаємодію інших його блоків. Це блок, який реалізує перехід до наступного рядка поточної підпрограми, або перехід до відпрацювання інших підпрограм, аварійних підпрограм, або підпрограм блокування видачі на ТА заборонених комбінацій команд керування;

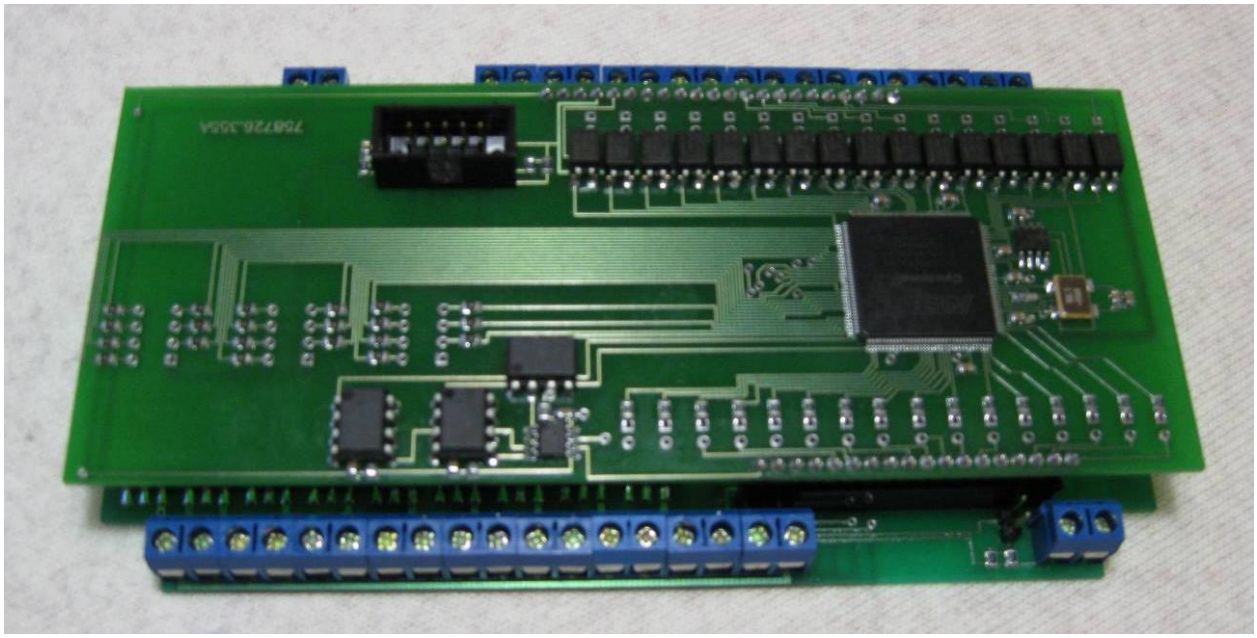


Рисунок 1.2 – Промисловий зразок ПЛІС-контролера паралельної дії

- ЛА – лічильник адреси. Призначений для формування адресації блоків пам’яті БПС, БПК та БПП до відповідного їх рядка, що визначається математичною моделлю роботи ЛКА ПД;

- ВР – вихідний регістр. призначений для передачі, та блокування, у випадку появи забороненої комбінації команд керування, сигналів керування виконавчими механізмами ТА.

Базову логіку роботи ПЛК ПД, як вже було вказано при перерахунку блоків, що його складають, формує блок логічного керування. БЛК, У блоці виконано реалізацію функції логічного керування за рахунок перевірки істинності рівнянь (1.18) і (1.19):

$$A = КП \vee ПР1 \vee ПР2, \tag{1.18}$$

$$+1 = E \wedge \overline{КП} \wedge \overline{ПР1} \wedge \overline{ПР2}, \tag{1.19}$$

де A – початкова адреса наступної підпрограми;

Е – сигнал еквівалентності вхідного вектора із очікуваним;

КП – ознака кінця відпрацювання поточної підпрограми;

ПР1 – ознака переривання від блоку пам’яті переходів;

ПР2 – ознака переривання від блоку пам’яті заборонених комбінації;

+1 – сигнал переходу до наступного рядка поточної підпрограми.

Для розуміння «механіки» роботи керуючого автомату, розглянемо технічну реалізацію рівнянь (1.18) та (1.19).

Вибір початкової адреси наступної підпрограми відбувається у випадку коли на вході блоку пам’яті переходів БПП сформується комбінація сигналів стохастичних входів, що співпадає з однією із запрограмованих. У такому випадку виконується частина рівності (1.18) *(КП=А)*. Це призводить до того, що БПП надає сигнал ЛА про встановлення шини адреси у відповідний даній комбінації стан. Наслідком цього стає формування блоком БЛК сигналу *«А»* – адреса який вказує блоку ЛА адресувати блоки пам’яті БПК і БПС на перший рядок обраної підпрограми. З технічної точки зору, для забезпечення можливості такого переходу, в останньому рядку кожної підпрограми, а також у нульовому рядку основного тіла програми записується тільки ознака «*КП*», яка ознака використовується у якості дозволу переходу ЛКА ПД до відпрацювання будь-якої із записаних у блоки пам'яті автомата набору підпрограм.

Для безпосереднього відпрацювання обраної підпрограми відбувається реалізація рівняння (1.19). Це відбувається наступним чином. У випадку виконання рівності (1.19) блок логічного керування формує сигнал «*+1*». Як результат появи цього сигналу, лічильник адреси ЛА виконує адресацію блоків пам’яті БПС і БПК на наступний, тобто (*i+1*) рядок поточної підпрограми.

Як було показано при розгляді математичної моделі роботи ЛКА ПД, формування сигналу «*Е*», може бути виконано двома способами:

- якщо на поточному кроці відпрацювання керуючої програми відбувається порівняння фактичного стану усіх датчиків циклу з їх

очікуваними значеннями, то в останній стовпчик *i*-го рядка, блоку пам'яті станів БПС, буде записано відповідну ознаку. Це призведе до того, що блоком вибору операцій БВО буде сформовано сигнал «*I=1*», тобто схему порівняння буде налаштовано на виконання логічної операції «*I*»;

- якщо для переходу до відпрацювання наступного рядка поточної підпрограми достатньо спрацювання лише одного з усієї кількості датчиків, то блоком БВО буде сформовано сигнал «*АБО=1*», що переведе схему порівняння на реалізацію логічної функції «*АБО*». У такому випадку сигнал еквівалентності *«E»* буде сформовано у разі збігу фактичного стану лише одного з датчиків циклу з їх очікуваними значеннями, записаними до і-го рядку блоку пам'яті станів.

Технічні сигнали *ПВ*» і «*БВ*», призначені для початкового встановлення ЛКА ПД при скиді та вмиканні та блокування видачі забороненої комбінації на виконавчі механізми. Ознаки переривань «*ПР1*», «*ПР2*», що входять до рівнянь (1.18) та (1.19) формують ознаки безмовного переходу до відпрацювання наступної підпрограми, без очікування відпрацювання поточної. Вони призначені для технічної реалізації безпекових функцій керуючого автомату.

Додаткову інформацію про роботу ЛКА ПД можна отримати у роботах [5, 14, 20].

1.2 Постановка завдання дослідження

Розглянутий вище матеріал показує, що керуючі пристрої з паралельною архітектурою мають дуже суттєві переваги у порівнянні з системами керування класичної послідовної дії. Серед таких переваг у першу чергу можна вказати дві найважливіших. Першою перевагою є практично повна відсутність залежності кількості контрольованих входів і керованих виходів від швидкодії контролера. Друга перевага – це можливість реалізації спеціалізованої TVP-технології, яка дозволяє створювати в

автоматизованому режимі і за допомогою спрощених мов, керуючі програми, неспеціалістом в області програмування. У літературних джерелах (наприклад [23]) було неодноразово вказано, що використання такої технології дозволяє значно зменшити кількість помилок у програмному коді та пришвидшити процес його написання. Ще одним, достатньо важливим аспектом, стає можливість уникнути непорозумінь між спеціалістом з технологічного процесу (технологом) і спеціалістом з програмування (програмістом). Таким чином удосконалення керуючих пристроїв з паралельною архітектурою для покращення їх характеристик, що дозволять більш широко застосовувати їх при створенні систем керування звичайними (промисловими, або навіть побутовими) об’єктами і процесами, є актуальною задачею.

Враховуючи вищевказане, основною метою роботи є розробка алгоритму функціонування керуючого автомату паралельної дії з розширеним функціоналом у вигляді програмованих таймерів.

Для досягнення поставленої мети дослідження потрібно розв’язати наступні часткові задачі дослідження:

- провести аналіз керуючих структур з паралельною архітектурою, розглянути їх внутрішню структуру, визначити шляхи та можливості вдосконалення;

- розглянути математичні моделі класичного керуючого автомату паралельної дії та ЛКА ПД з розширеними функціональними можливостями, що реалізують програмовані таймери;

- дослідити універсальний алгоритм функціонування ЛКА ПД;

- визначити шляхи інтеграції до існуючого алгоритму елементів, що дозволять реалізувати додатковий функціонал;

- розробити алгоритм функціонування вдосконаленого ЛКА ПД, що містіть у собі елементи для реалізації програмованих таймерів.

2 ОСОБЛИВОСТІ ПОБУДОВИ ЛКА ПД ІЗ ФУНКЦІЯМИ ВБУДОВАНИХ ПРОГРАМОВАНИХ ТАЙМЕРІВ

2.1 Архітектура ЛКА ПД у категоріях математичної моделі

Розглянута у попередньому розділі структура класичного ЛКА ПД орієнтована на викладення принципів його роботи з точки зору практичної реалізації контролера на базі сучасних ІМС – кристалів ПЛІС. Саме тому елементи цієї структури – блоки, містять опис і пояснення функціонування у категоріях сигналів, елементів, блоків пам’яті, тощо. У той же час кінцевою метою цієї роботи є розробка алгоритму функціонування автомату, тож розглянемо структуру ЛКА ПД у категоріях її математичної моделі: матрицях, функціях, векторах. Структуру такого типу показано на рисунку 2.1

Основу структури ЛУА ПД становлять чотири матриці: $|M_A|$ – очікуваних станів детермінованих входів (матриця станів); $|M_B|$ – очікуваних станів стохастичних входів (матриця адрес переходів); $|M_C|$ – матриця команд керування та матриця $|M_E|$ – матриця заборонених станів. Призначення цих матриць було наведено вище.

Принцип дії автомата практично повністю повторює роботу ЛКА ПД по щойно розглянутому формальному опису у розділі 1.1.2 роботи. Але для більш повного розуміння і більш коректного переходу до розгляду алгоритма функціонування ЛКА ПД вкажемо відповідність розглядаємої структури, математичної моделі і структури з попереднього розділу.

Окрім вже згаданих блоків пам’яті на структурі присутні:

- схема порівняння СП – позначено символом α_p має своє відображення у виразі (1.14) математичної моделі;

- блок індикації БІ, що визначений на структурі, як відсутність тотожності вхідного вектору $a_j(t_p)$, запрограмованому A_{ij};

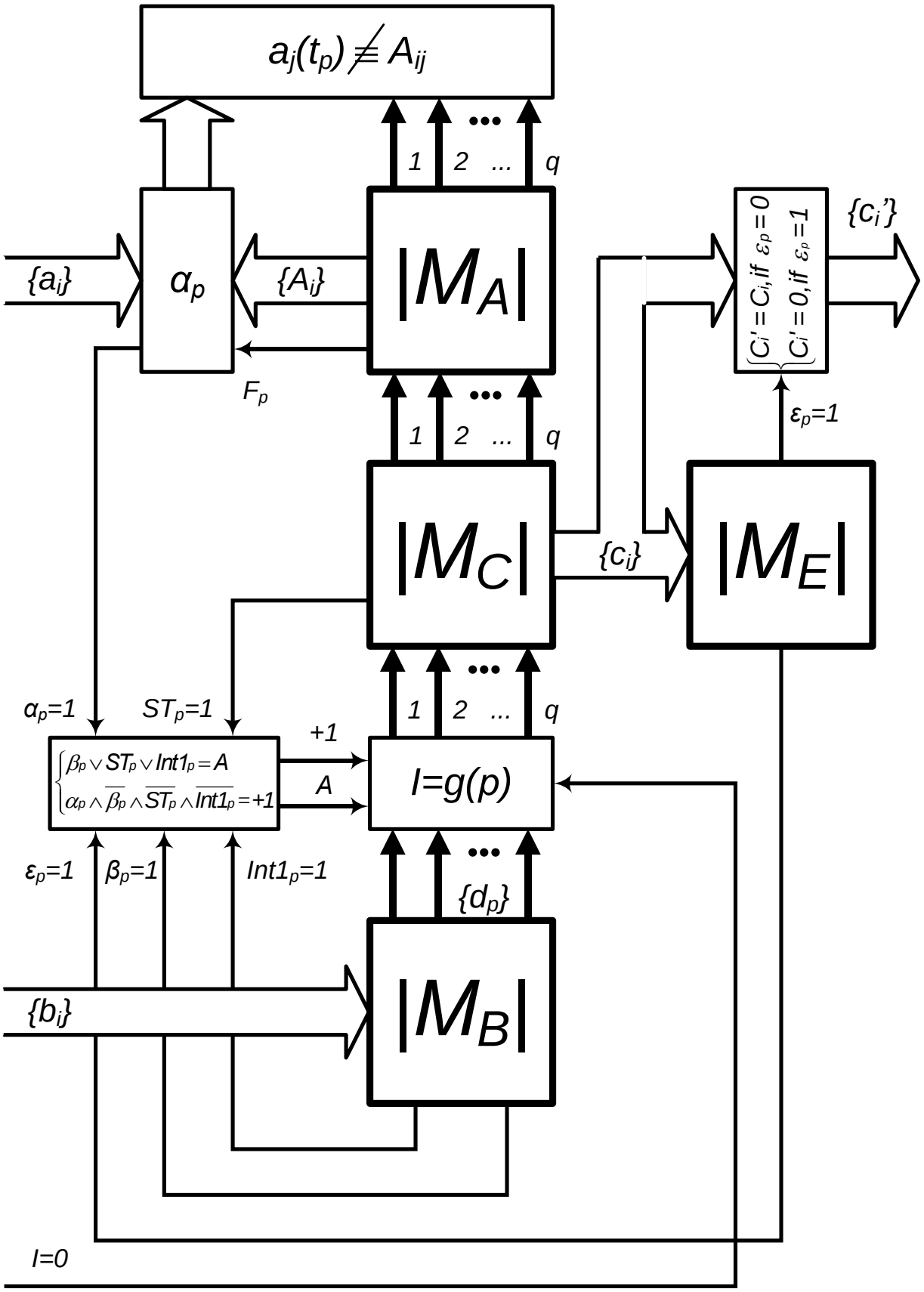


Рисунок 2.1 – Структура логічного керуючого автомату паралельної дії

- блок логічного керування БЛК – на структурі позначено системою рівнянь (1.18), (1.19) але у термінах математичної моделі4

- лічильник адреси ЛА. Отримав на структурі позначення відповідної функції математичної моделі – $I = g(p)$;

- вихідний регістр ВР показано, як систему рівнянь, що блокує видачу вектора керувань $\{c_i'\}$ на виконавчі механізми.

Відповідність усіх інших сигналів і векторів математичної моделі легко прослідкувати, порівнянням структур на рисунках 1.1 і 2.1.

2.2 Структура керуючих автоматів із функціями вбудованих програмованих таймерів

Розгляд структури і математичної моделі керуючого автомату паралельної дії, що був проведений вище, показав, що, для реалізації функцій програмованих таймерів в таких пристроях, необхідно вносити зміни до рівнянь (1.18) і (1.19), які власне і визначають базову логіку роботи автомату. З цього можна зробити висновок, що і у випадку переходу на наступний рядок поточної підпрограми. і для переходу до відпрацювання іншої підпрограми мають бути змінені обидва рівняння, що визначають роботу автомату.

Таким чином для реалізації функцій програмованих таймерів необхідно до деяких елементів структури ЛКА ПД (рисунок 1.1) внести певні зміни. Також необхідно змінити деякі зв’язки, що приймають участь, у формуванні сигналів «*+1*» і «*КП*».

На рисунку 2.2 показано структуру ЛКА паралельної дії із розширеним функціональними можливостями у якій зроблено спробу реалізації функції внутрішніх програмованих користувачем таймерів [9].

Розглянемо відмінності та особливості функціонування вдосконаленої структури, що відбулись у зв’язку з додаванням елементів, що реалізують функціонал внутрішніх програмованих таймерів.

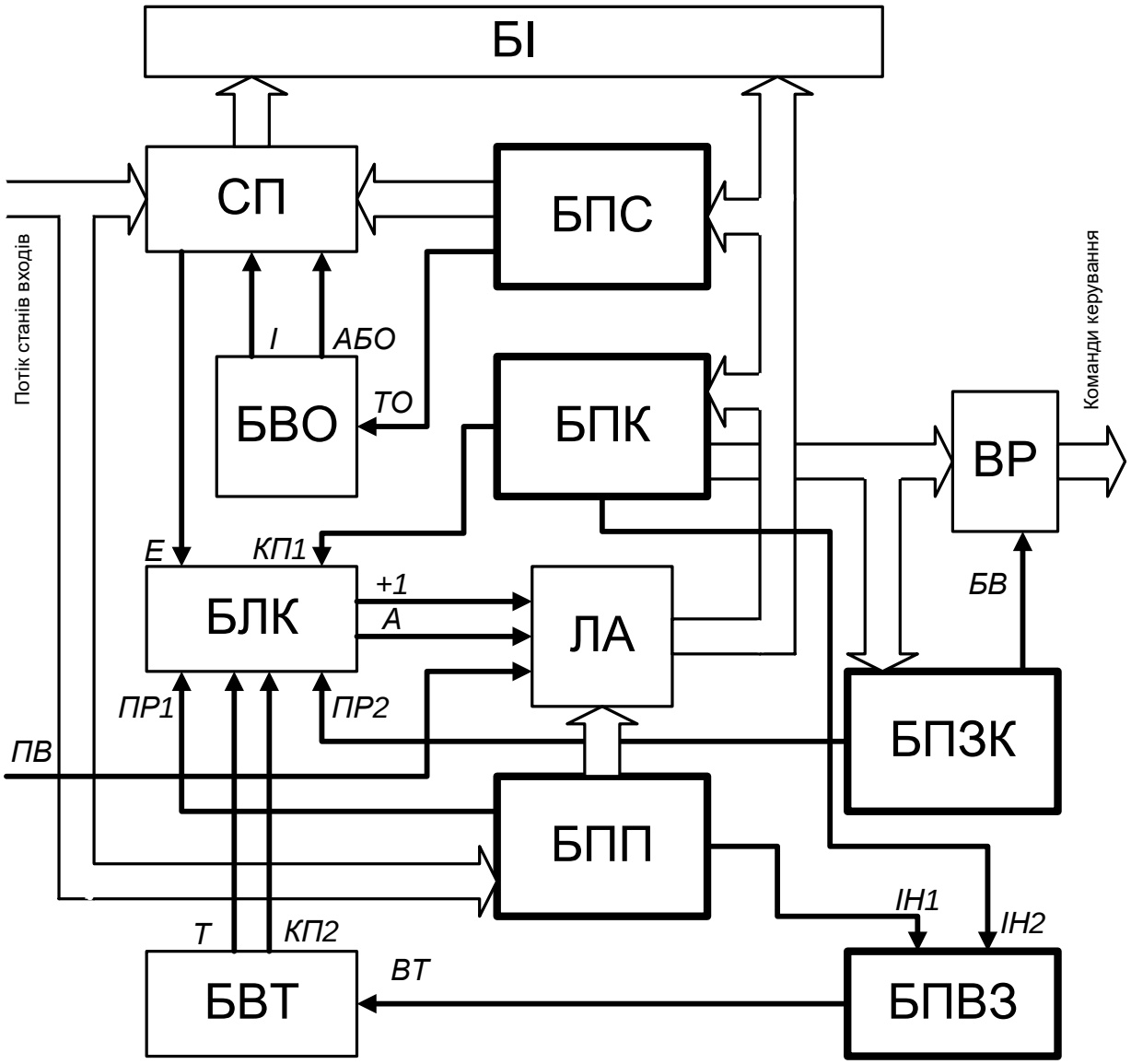


Рисунок 2.2 – Структура ЛКА ПД з елементами реалізації внутрішніх програмованих таймерів

Основною відмінністю такої структури є уведення до неї додаткового блоку пам’яті – блоку пам’яті внутрішніх змінних (БПВЗ). Цей блок пам’яті містить чисельні значення відліків внутрішніх таймерів. що програмуються користувачем. Ще одним додатковим блоком став блок внутрішніх таймерів – БВТ. Блок БВТ, разом з блоком БЛК формують загальну логіку роботи ЛКА ПД, причому БВТ спеціально не був інтегрований до загального блоку логіки БЛК, для того, щоб спростити обґрунтування роботи програмованих таймерів та візуально виділити елементи та зв’язки між ними, що формують

таку логіку.

Отже можна переходити до опису призначення додаткових сигналів і блоків, що відрізняють вдосконалену структуру від базової.

Про блок пам’яті внутрішніх змінних вже було згадано, фактично цей блок аналогічний по своїй структурі блоку БПЗК. У математичній моделі це відображується появою вектора *iv (internal variable)* – вектора внутрішніх змінних. Цей вектор являє собою чисельне значення проміжку часу, що формується таймером. Кількість таких векторів визначається кількістю програмованих таймерів, а їх розрядність – довжиною проміжку часу, що формується і, у загальному випадку дорівнює $w \times v$, де w – кількість програмованих таймерів, а v – розрядність відповідних таймерів. Сукупність векторів *iv* формує матрицю *VR* – внутрішніх змінних.

Основними додатковими сигналами оновленої структури є сигнали ініціатори запуску відліку внутрішніх таймерів – «*IH1*» та «*IH2*». Фактично ознаки цих сигналів записано до *i*-го рядку блоку пам’яті команд, або блоку пам’яті переходів у вигляді елемента відповідного стовпчика, аналогічно, до ознаки *«КП»*, що розглядалось раніше. Ці сигнали ініціатори по суті свого впливу на процес формування сигналів «*+1*» та *«А»*, аналогічні сигналам від відповідних датчиків, детермінованих – для БПК, або стохастичних – для БПП.

У момент часу. що збігається з моментом часу закінчення формування інтервалу таймером, блоком пам’яті внутрішніх змінних формується відповідний сигнал «*BT*». Цей сигнал має наступне логічне значення: внутрішній таймер завершив свою роботу і необхідний проміжок часу сформовано. За наявності сигналу «*BT*» блок внутрішніх таймерів ініціює формування:

- сигналу «*T*», якщо ініціатором процесу формування відліку був БПК;

- сигналу «*КП2*», якщо ініціатором був БПП.

Сигнал «*T*» по своїй суті аналогічний сигналу *«Е»* схеми порівняння. Він надає інформацію блоку БЛК про те, що необхідно сформувати сигнал

«*+1*», тобто виконати перехід до відпрацювання наступного *i+1*-го рядка поточної підпрограми. Тобто логіка роботи автомата з внутрішнім таймером у частині, що відповідає за процес покрокового відпрацювання поточної підпрограми, фактично відповідає логіці обробки сигналів детермінованих входів:

$$+1 = (E \vee T) \wedge \overline{КП1} \wedge \overline{КП2} \wedge \overline{ПР1} \wedge \overline{ПР2} \qquad (2.1)$$

Логіка роботи автомата при переході першого рядка іншої підпрограми наступна. У цьому випадку ініціатором процесу відліку інтервалу часу є блок БПП і, відповідно, блок блоком внутрішніх таймерів формується вже сигнал «*КП2*». а не «*Т*». Логіка роботи з сигналом «*КП2*» повністю відповідає логіці роботи з сигналом «*КП*» (кінець підпрограми) класичної структури. Але необхідно зазначити. що у розглядаємій структурі сигнал «*КП*» отримав іншу назву. а точніше додатковий індекс – «*КП1*». Це пов’язано, як вже було вказано раніше, для полегшення формування і опису логіки роботи автомата. Таким чином перехід до наступної підпрограми відбувається лише по закінченню сформованого внутрішнім таймером відліку часу. Такий алгоритм роботи з підпрограмами і внутрішніми таймерами одночасно, повністю аналогічний до ситуації, коли ознака кінця підпрограми фіксується БПК у звичайних умовах, тобто без участі внутрішніх програмованих таймерів. Зазначимо, що мова йде про роботу зі звичайними підпрограмами. Робота з аварійними підпрограмами, що ініціюються перериваннями *«ПР1»* і *«ПР2»* не змінюється, у порівнянні з класичною структурою.

Таким чином формування сигналу «*А*» буде визначатись наступним виразом:

$$A = КП1 \vee КП2 \vee ПР1 \vee ПР2 \qquad (2.2)$$

Якщо перейти до категорій математичної моделі, то у підсумку необхідно зафіксувати відповідність виразу (2.1) виразу (2.3), а також виразу (2.2), виразу (2.4).

$$I=g(p)=g(p-1)+1, \text{ якщо } (ST_p \wedge Int1_p \wedge \alpha_p \wedge vr_p)=0. \quad (2.3)$$

$$I=g(p)=[dp=f(bp)], \text{ якщо } [(b_p \wedge ST_p) \vee Int1_p=1] \wedge (\alpha_p \wedge vr_p)=0\ . \quad (2.4)$$

Вкажемо також, що процедура формування проміжку часу таймером є незалежною від усіх інших процесів, що протікають в ПЛК ПД і виконується паралельною з іншими процесами, що відбуваються у ньому. Тобто паралельний принцип роботи автомата ніяк не порушується. Реалізація процедури відліку виконується незалежними апаратними засобами кристалу ПЛІС, тобто виконання процедури відліку проміжку часу програмованим таймером ніяким чином не залежить від внутрішньої логіки роботи класичного ЛКА ПД.

3 АЛГОРИТМ ФУНКЦІОНУВАННЯ КЕРУЮЧИХ АВТОМАТІВ З ПАРАЛЕЛЬНОЮ АРХІТЕКТУРОЮ

3.1 Підходи до паралельного програмування і побудови паралельних алгоритмів

Паралельне програмування вже багато років використовується у різних галузях, від наукових досліджень до бізнес-додатків та ігрової індустрії. Його застосування дозволяє прискорити обчислення, обробку великих обсягів даних та покращити продуктивність програмних систем. Розглянемо основні принципи, переваги та недоліки, що можуть бути застосовані у паралельному програмуванні та побудови паралельних алгоритмів.

Отже паралельне програмування – це процес поділу задачі на безліч дрібніших і незалежних підзадач, які можуть бути виконані одночасно на різних обчислювальних пристроях, або складових частинах пристрою. Таким чином, прискорюється час виконання та підвищується продуктивність програми.

Основною метою паралельного програмування є прискорення роботи програм та обробки даних. Паралельне програмування активно використовується в областях, де необхідно обробляти великі обсяги даних або обчислення, повинні бути виконані в найкоротші терміни (рисунок 3.1).

Розглянемо підходи до паралельного програмування. При розробці паралельних програм існує кілька підходів, що можна використовувати для розв’язання завдань. Розглянемо основні з них:

- багатопотокове програмування. Цей підхід використовує кілька потоків виконання, що працюють паралельно та виконують завдання, що дозволяє збільшити продуктивність програми за рахунок використання кількох ядер процесора;

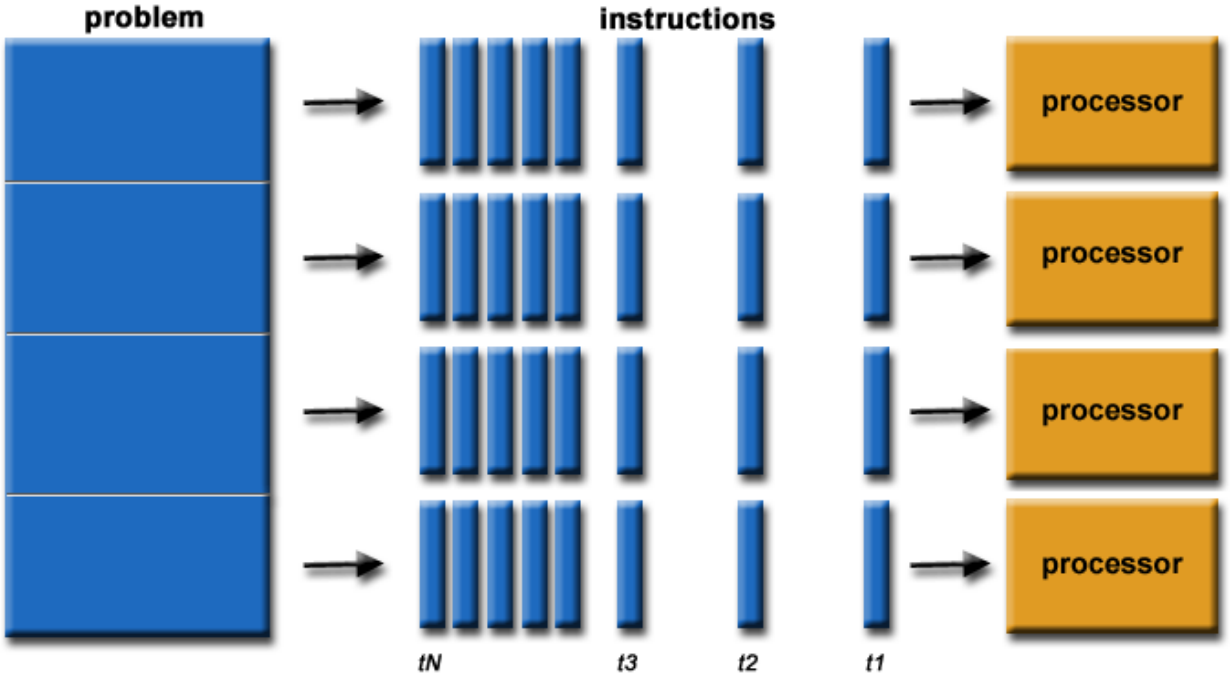


Рисунок 3.1 – Схематичне представлення процедури паралельного програмування

- розподілене програмування. У цьому підході завдання розподіляються між кількома комп'ютерами, що працюють у мережі. Це дозволяє розв’язувати завдання більшого обсягу та прискорювати виконання обчислень;

- асинхронне програмування. При використанні цього підходу потоки виконання не блокуються на очікуванні завершення завдання, а продовжують роботу над іншими завданнями. Це дозволяє значно збільшити ефективність використання ресурсів та зменшити час очікування;

- функціональне програмування. У цьому підході використовуються функції, що не мають стану і не змінюють зовнішні змінні. Це робить програми більш надійними та передбачуваними;

- GPU-програмування. Цей підхід використовує графічні процесори для виконання завдань. Це дозволяє прискорювати виконання операцій, пов’язаних із графікою та обробкою зображень.

Залежно від поставленої задачі та використовуваних технологій можна

вибрати підхід, який найбільше підходить для вирішення конкретної задачі.

Застосування паралельного програмування є важливим інструментом у різних галузях, таких як наука, бізнес та ігрова індустрія. Це дозволяє ефективно використовувати ресурси обчислювальних і керуючих систем та прискорити виконання завдань.

Вкажемо декілька прикладів використання паралельного програмування у різних галузях:

- у науці паралельне програмування використовується для обробки великих обсягів даних, таких як моделювання клімату, генетичних даних та астрономічних спостережень;

- у бізнесі паралельне програмування може бути використане для прискорення обчислень у сфері фінансів, банківської справи, маркетингу та торгівлі;

- в ігровій індустрії паралельне програмування дозволяє створювати більш складні та реалістичні ігри, що можуть опрацьовувати велику кількість інформації та взаємодій між гравцями.

Роль паралельного програмування в обробці великих обсягів даних та прискоренні обчислень полягає у можливості розподілу роботи між кількома ядрами процесора. Це дозволяє прискорити виконання завдань, які можуть бути виконані паралельно, такі як обробка великих обсягів даних або виконання складних обчислювальних завдань. Паралельне програмування також дозволяє використовувати ресурси обчислювального або керуючого пристрою ефективніше, що призводить до суттєвого прискорення виконання завдань.

Наведемо переваги та недоліки використання паралельного програмування.

Почнемо з переваг:

- збільшення продуктивності та швидкості обробки даних за рахунок використання багатопоточності та розподілених обчислень;

- зменшення часу відповіді та підвищення чуйності системи, що

особливо важливо для програм, систем і пристроїв. що працюють у реальному масштабі часу;

- можливість обробки великих обсягів даних та виконання складних обчислень, які можуть бути неможливими у у випадку використання послідовного принципу роботи.

Недоліки:

- складність налагодження та тестування паралельних програм, що пов’язано з потенційними проблемами синхронізації даних та конкуренцією за ресурси;

- необхідність знання специфічних технологій та алгоритмів для ефективного використання паралельного програмування;

- ускладнення структури програми і можливість появи помилок при роботі з даними та ресурсами.

Як видно, переваги паралельного програмування, такі як збільшення продуктивності та зниження часу відповіді, роблять його важливим інструментом у розробці програмного забезпечення. Однак, розробники повинні враховувати можливі складнощі, такі як проблеми синхронізації даних та вимоги додаткових зусиль та знань.

3.2 Алгоритм функціонування керуючого автомата паралельної дії

Як же було показано, технології паралельного програмування і формування паралельних алгоритмів – це важлива технологія, що може підвищити продуктивність та прискорити обробку великих обсягів даних. Це може бути особливо корисним для проектів, пов’язаних з науковими дослідженнями, машинним навчанням, аналізом даних та іншими обчислювально-інтенсивними завданнями, побудовою систем керування складними або відповідальними об’єктами. Однак, цей процес також може бути складним і вимагати врахування багатьох факторів, включаючи проблеми синхронізації даних та складності налагодження. Використання

правильних моделей паралельного програмування, адекватних підходів та мов програмування може допомогти впоратися з цими проблемами та отримати максимальну вигоду від паралельних обчислень та паралельного керування.

У відповідності до математичної моделі, що була розглянута раніше, алгоритм функціонування керуючого автомата паралельної дії може бути описаний за допомогою достатньо великой кількості підходів, технологій та мов. Але всі ці технології так або інакше орієнтовані на використання у якості обчислювального або керуючого ядра пристроїв послідовної дії. Ці підходи не можуть бути ефективно використані для складання алгоритмів функціонування автоматів з паралельним принципом функціонування. Це пов’язано, в основному, з тим, що у процесі роботи необхідно реалізовувати велику кількість логічних операцій та операцій присвоєння, які мають виконуватись асинхронно та незалежно одна від одної. тобто виконуватись виключно паралельно.

За класичним визначенням, паралельний алгоритм, що протиставляється традиційним послідовним алгоритмам, – це алгоритм, що може бути реалізований частинами на безлічі різних обчислювальних пристроїв з подальшим об’єднанням отриманих результатів та отриманням коректного результату.

Паралельні алгоритми дуже важливі через постійне вдосконалення багатопроцесорних систем та збільшення числа ядер у сучасних процесорах. Зазвичай простіше сконструювати обчислювальний пристрій із одним швидким процесором, ніж із безліччю повільних процесорів (за умови досягнення однакової продуктивності). Однак продуктивність процесорів збільшується головним чином за рахунок удосконалення техпроцесу (зменшення норм виробництва), чому заважають фізичні обмеження на розмір елементів мікросхем та тепловиділення. Зазначені обмеження можуть бути подолано шляхом початку багатопроцесорної обробки, або, фактично, розпаралелювання, що виявляється ефективним навіть малих

обчислювальних або керуючих систем.

Для формального описання паралельних алгоритмів і послідовно-паралельних алгоритмів, тобто послідовних алгоритмів із паралельними ділянками, використовувалось достатньо багато різних підходів, методів і технологій. Згадаємо основні з них [18]:

- метод ярусно-паралельних форм (ЯПФ);

- мова операторних схем паралельних алгоритмів із пам’яттю (ОСПАП);

- метод паралельних гілок;

- метод паралельних операторів;

- використання мереж Петрі.

Усі вони характеризуються більшою, або меншою ефективністю, але найбільш зручним виявився метод паралельних гілок, з деякими удосконаленнями. Він надає найбільш наглядне, зручне і просте уявлення про роботу системи і прийняті рішення.

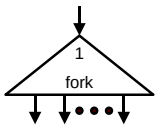
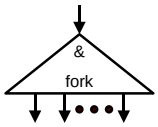
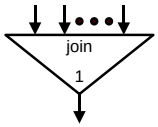
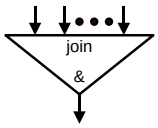
Розглянемо базову сутність базового методу паралельних гілок, що полягає у наступному. Паралельним оператором будемо називати сегмент програми, що складається із паралельних гілок, що реалізуються незалежно одна від одної. Це фрагменти програми, якими можуть бути окремі оператори або послідовності операторів. У тому числі такими фрагментами програми можуть бути і її паралельні сегменти. Сегмент обов’язково повинен мати початок і кінець. Початком будемо називати точку розгалуження гілок. а її кінцем – точку їх об’єднання. У процесі реалізації алгоритму логічна обробка інформації у кожній із паралельних гілок виконується незалежно від інших до кінця сегменту. Після закінчення процедури обробки виконання призупиняється і знаходиться у режимі очікування, до тих пір поки не закінчиться логічна обробка інших паралельних гілок.

У базового методу паралельних гілок присутній дуже суттєвий недолік – наявність лише кон’юнктивної форми розходження і злиття паралельних гілок. За рахунок використання його удосконаленої версії. з’явилась

можливість використання принципу «необхідної і (або) достатньої інформації», тобто інформації необхідної і (або) достатньої для реалізації паралельних ділянок. Цей підхід дозволяє реалізувати як кон'юнктивну, так і диз'юнктивну форми розходження і злиття гілок, оскільки при описі алгоритму функціонування логічного керуючого автомата паралельної дії, використовуються обидві форми.

Усі варіанти можливих видів і форм розгалужень, а також терміни і умовні позначення, що складають сутність удосконаленого методу паралельних гілок, зведені до таблиці 3.1.

Таблиця 3.1 – Удосконалені форми розходження і злиття паралельних гілок

Вид розгалуження	Форма розгалуження	Уживаний термін		Умовне позначення
		Англійський	Український	
Розходження гілок	диз'юнктивна	fork-or	розгалуження за «АБО»	
	кон'юнктивна	fork-and	розгалуження за «І»	
Злиття гілок	диз'юнктивна	join-or	злиття за «АБО»	
	кон'юнктивна	join-and	злиття за «І»	

Розглянемо технологію побудови алгоритму функціонування класичного керуючого автомата паралельної дії. Для значного спрощення такої процедури процес розробки алгоритму розбивається на два етапи. На першому етапі необхідно скласти повну множина макрооператорів паралельних гілок. На другому з складених макрооператорів будується

загальна блок-схема алгоритму.

У відповідності до раніше розглянутої математичної моделі і структури керуючого автомата функціонально повна множина макрооператорів має містити:

- макрооператор початкового встановлення;

- макрооператор формування вектора керувань виконавчими механізмами;

- макрооператор аналізу заборонених комбінацій команд керування виконавчими механізмами;

- макрооператори аналізу сигналів детермінованих входів (при цьому аналіз має відбуватись як за логікою «І», так і за «АБО»);

- макрооператор аналізу сигналів стохастичних входів.

Розглянемо блок-схеми перерахованих вище макрооператорів та їхні спрощені умовні позначення, які будуть використані при формуванні узагальненої блок-схеми алгоритму.

Макрооператор початкового встановлення (рисунок 3.2) забезпечує встановлення внутрішніх сигналів. таких як крок, адреса, кінець підпрограми та безумовне переривання, у відповідний початковий стан. Він реалізує розпаралелювання операцій типу присвоєння.

Макрооператор формування вектора керувань, що показаний на рисунку 3.3 також призначений для реалізації розпаралелювання операцій типу присвоєння. але вже при формуванні вихідних сигналів керування виконавчими механізмами технологічного агрегату.

Макрооператори аналізу станів детермінованих входів (рисунок 3.4 і 3.5) призначені для поелементного паралельного і незалежного логічного порівняння вектора фактичних станів керованого об’єкта a_j, із їх значеннями, що запрограмовано у матриці станів $|M_A|$. У залежності від того, яким чином формується перехід до наступного кроку алгоритму, може бути використано два підходи. При умові співпадіння усіх фактичних станів датчиків циклу з їх очікуваними значеннями має бути використана логічна операція «І» (рисунок

3.4), а при умові наявності сигналу хоча б від одного з датчиків – реалізується порівняння за логікою«АБО» (рисунок 3.5). Обидва вказаних макрооператори порівняння призначені для розпаралелювання логічних операцій.

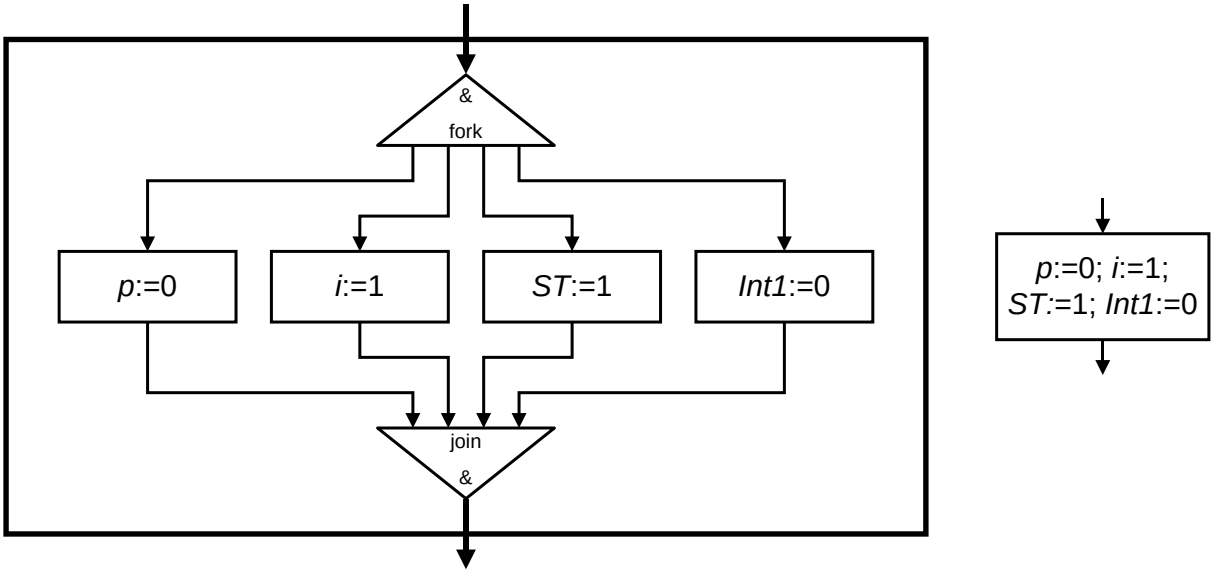


Рисунок 3.2 – Макрооператор початкового встановлення

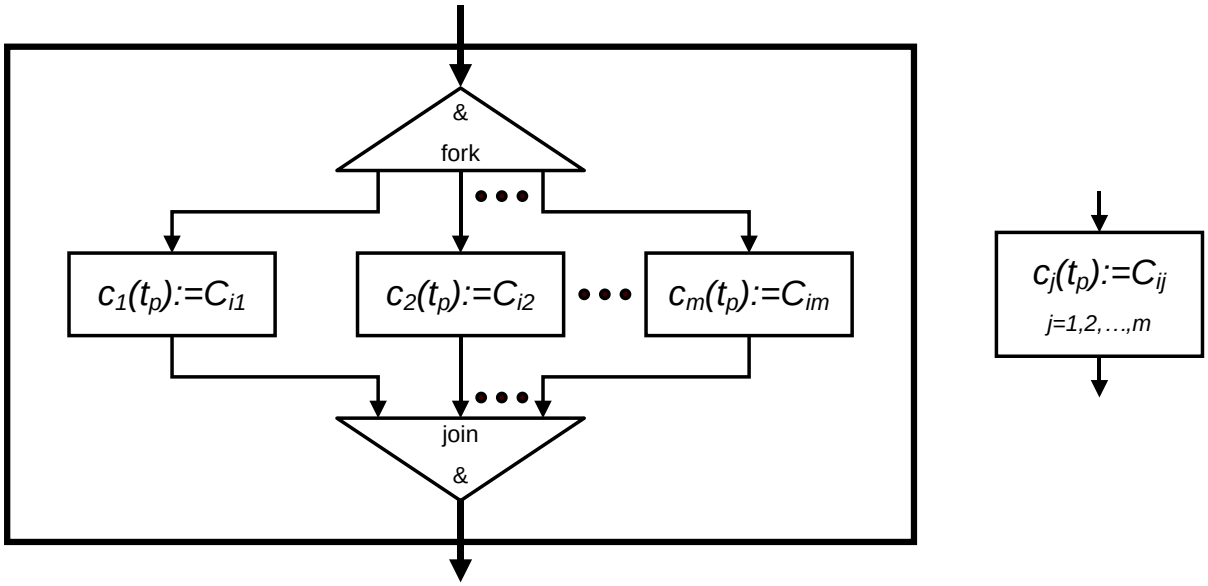


Рисунок 3.3 – Макрооператор формування вектора керувань

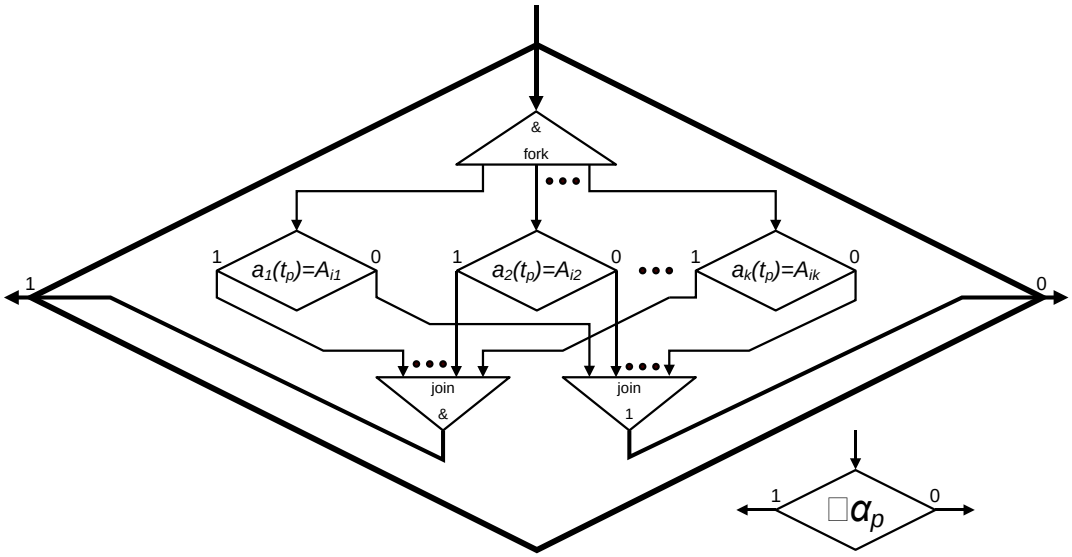


Рисунок 3.4 – Макрооператор аналізу станів детермінованих входів за логікою «І»

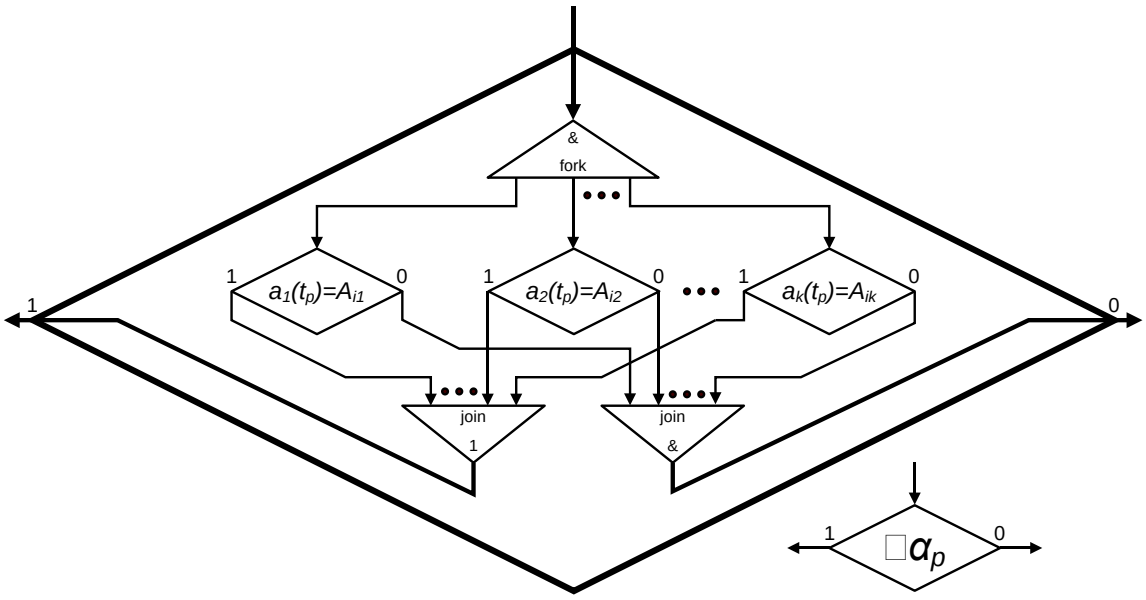


Рисунок 3.5 –Макрооператор аналізу станів детермінованих входів за логікою «АБО»

Макрооператор, що показано на рисунку 3.6, призначений для унеможливлення формування комбінації вихідних команд керування, що можуть призвести до аварії на технологічному об’єкті – макрооператор аналізу заборонених комбінацій команд керування. Призначений для виконання паралельного поелементного порівняння компонент вектора c_j, зі значенням компонент векторів e_j, що записані до матриці $|M_E|$. При цьому паралельно і незалежно перевіряється факт співпадіння із вектором команд керування виконавчими механізмами технологічного агрегату, кожного рядка матриці заборонених станів $|M_E|$. У випадку наявності хоча б одного співпадіння, формується сигнал логічної одиниці на виході макрооператору, і видача такої комбінації на виконавчі механізми блокується. Розглянутий макрооператор, як і два попередніх, також виконує функції розпаралелювання логічних операцій.

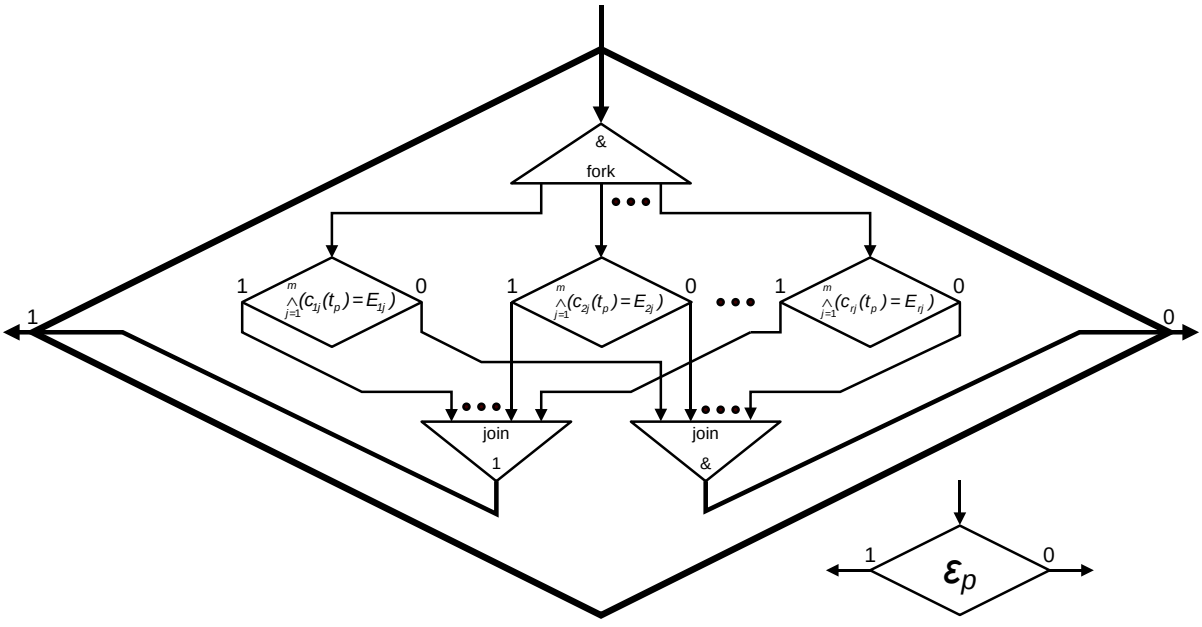


Рисунок 3.6 –Макрооператор аналізу заборонених комбінацій команд керування технологічним обладнанням

Макрооператор аналізу сигналів стохастичних входів (рисунок 3.7) призначений для формування адреси переходу у програмах розгалужених з розгалуженнями, він так само виконує функції розпаралелювання логічних операцій. Робота макрооператора полягає у паралельному, незалежному порівнянні компонент вектора стану стохастичних входів b_j з умовами, що запрограмовані у матриці $|M_B|$. Тобто паралельно і незалежно перевіряється співпадіння кожного рядка матриці умов переходів $|M_B|$ із вектором станів стохастичних входів. У випадку співпадіння комбінацій, формується сигнал логічної одиниці на виході макрооператора, що є ознакою для визначення і формування наступної підпрограми.

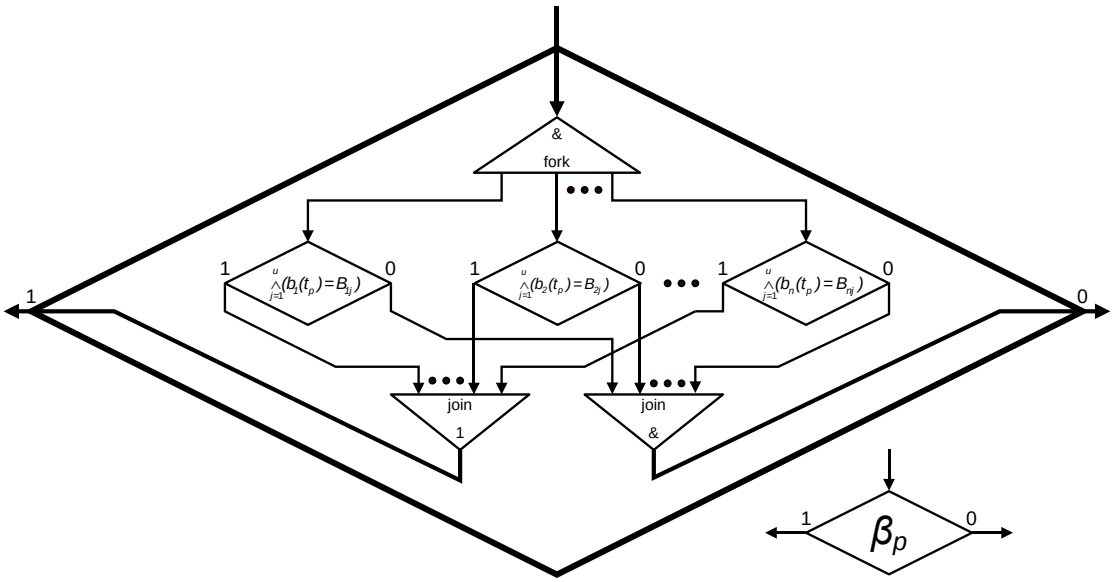


Рисунок 3.7 – Макрооператор аналізу сигналів стохастичних входів

Блок-схему універсального алгоритму функціонування логічного керуючого автомату паралельної дії показано на рисунку 3.8. Його складено як результат поєднання розглянутих раніше макрооператорів, а також інших логічних і функціональних і операторів, що виконують допоміжні функції.

Слід окремо зазначити, що у показаній блок-схемі, макрооператори розглядаються як звичайні, тобто нерозгалужені функціональні і логічні

оператори.

З наведеного алгоритму чітко видно, що при роботі ЛКА ПД відбувається паралельна і незалежна у трьох його гілках: обробка сигналів детермінованих входів, обробка сигналів стохастичних входів і обробка безумовних переривань, що призводять до переходу до відпрацювання аварійних програм.

Більш детально з роботою наведеного на рисунку 3.8 алгоритму можна ознайомитись у [5, 22].

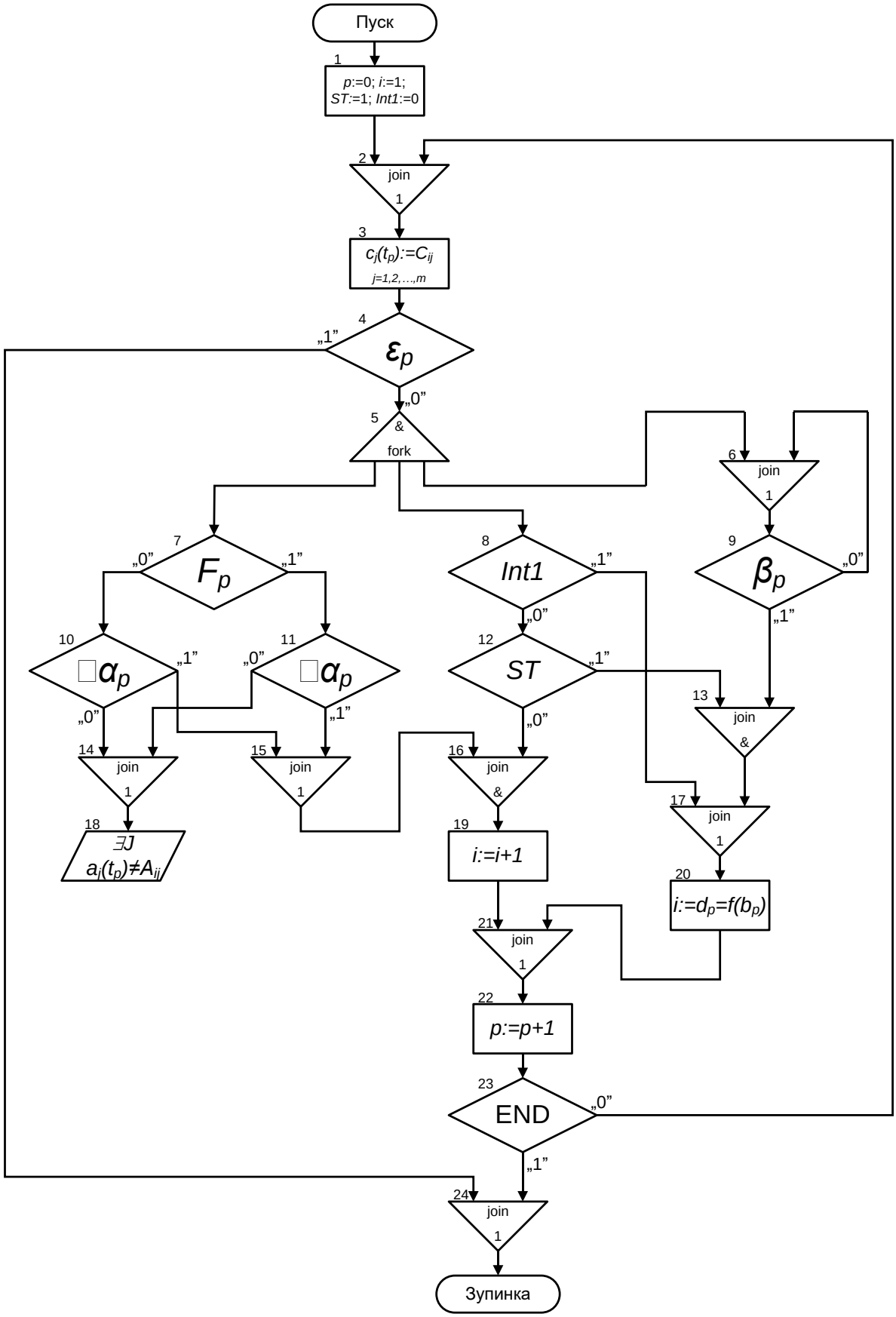


Рисунок 3.8 – Універсальний алгоритм функціонування класичного ЛКА ПД

4 РОЗРОБКА АЛГОРИТМУ ФУНКЦІОНУВАННЯ ПЕРСПЕКТИВНОГО ЛКА ПД

З урахуванням тих змін, що зазнали математична модель і структура керуючого автомата паралельної дії у частині реалізації програмованих користувачем таймерів, алгоритм його функціонування може бути представлений блок-схемою, що наведено на рисунку 4.1. Покроково розглянемо роботу вдосконаленого ЛКА ПД.

На початку роботи (у момент пуску технологічного обладнання або скиду контролера) відбувається початкове встановлення кроку – *p*, рядка програми – *i*, переривання – *Int1* і дозволу переходу до підпрограми – *ST*. За цю процедуру відповідає макрооператор 1.

На наступному етапі за допомогою макрооператора 3 відбувається формування вектора керувань виконавчими механізмами технологічного агрегату $c_{ij}(t_p):=C_{ij}$. Цей вектор, перш ніж бути перетвореним у фізичні сигнали керування виконавчими механізмами, перевіряється макрооператором 4 на наявність на виході автомата забороненої комбінації команд керування. У випадку наявності забороненої комбінації, ним формується сигнал логічної одиниці і робота обладнання блокується до усунення причин її появи.

У випадку відсутності заборонених комбінацій, сигнали керування виконавчими механізмами змінюють стан керованого об’єкта, що має бути проконтрольовано системою відповідних датчиків. На цьому етапі починається процес паралельного одночасного аналізу вектора детермінованих (макрооператори 13 і 14) і стохастичних (макрооператор 10) входів. Одночасно і паралельно з цим процесом відбувається формування відліку вбудованим програмованим таймером, звісно, якщо на поточному кроці формування відліку часу передбачено алгоритмом керування технологічним обладнанням. за цей процес відповідають оператори 7, 11, 12.

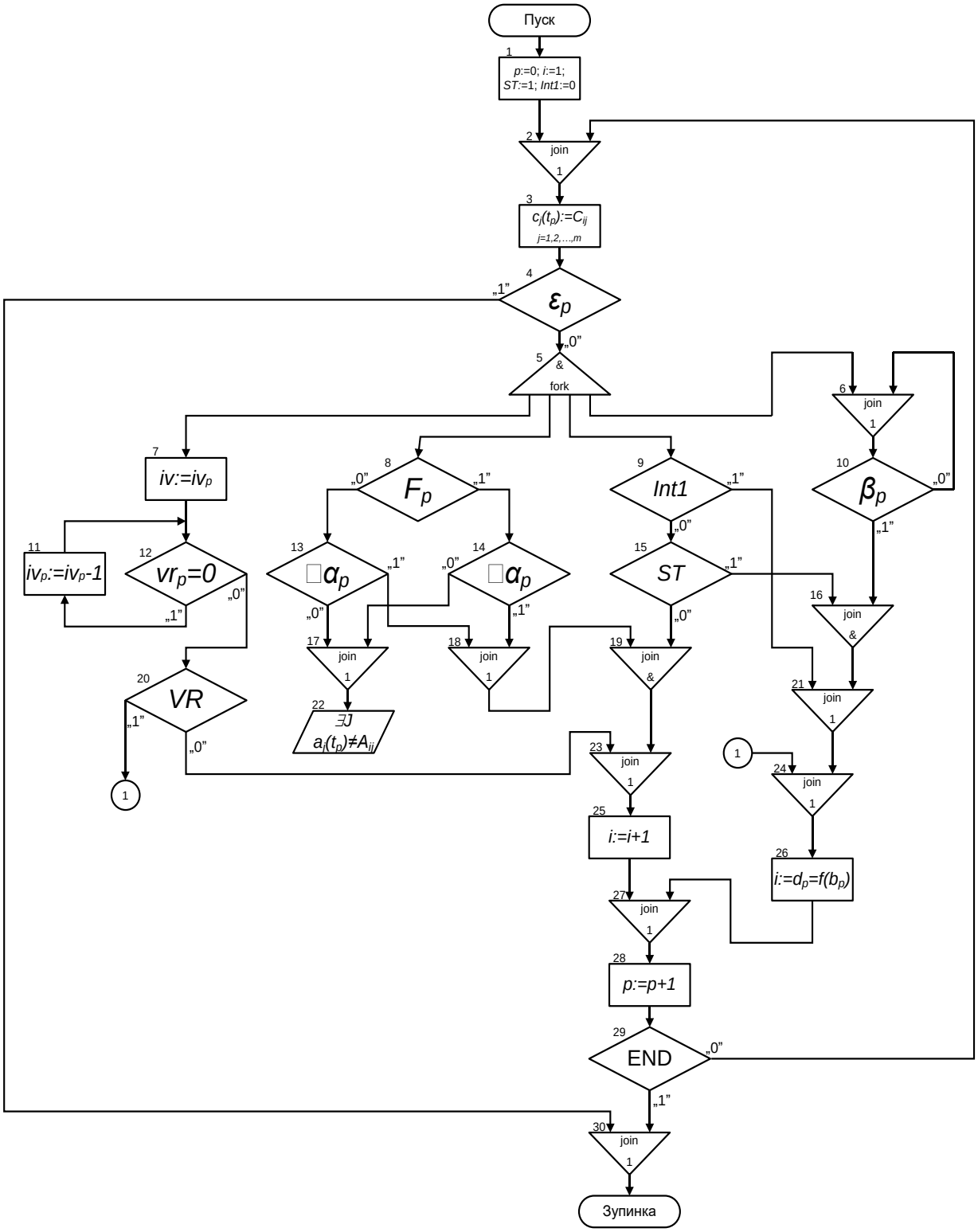


Рисунок 4.1 – Алгоритм функціонування вдосконаленого ЛКА ПД

Ще однією гілкою паралельного алгоритму є реалізація процесу перевірки наявності переривання *Int1* і дозволу переходу *ST*. У залежності від результатів цієї перевірки відбувається запуск процедури переходу то того, або іншого кроку керуючої програми.

У випадку наявності аварійного переривання *Int1*, виконується умова оператора 9, і лічильник адреси формує початковий рядок адреси підпрограми обробки аварійного переривання. Перехід до відпрацювання цієї підпрограми вказує на ненормальний хід технологічного процесу (це може бути, наприклад, підпрограма аварійної зупинки технологічного процесу кнопкою «СТОП»). Після усунення джерела появи аварійного переривання хід нормального відпрацювання алгоритму керування обладнанням поновлюється.

Слід окремо зупинитись на різниці між відпрацюванням аварійної підпрограми і підпрограми обробки заборонених комбінацій команд керування. Аварійні підпрограми – це підпрограми, що передбачені алгоритмом керування технологічним обладнанням. Замість терміна «аварійні», краще було б застосувати такі як «невідкладні» або «термінові» підпрограми. Але у середовищі технологів за такими програмами стійко закріпився саме термін «аварійні». Зовсім інший випадок – відпрацювання підпрограми, що блокує видачу забороненої комбінації команд керування. Видача такої комбінації на виконавчі механізми однозначно призведе до аварії на технологічному об’єкті, тому її має бути заблоковано, а роботу обладнання зупинено до моменту з’ясування причин появи такої комбінації. Прикладом такої забороненої комбінації може бути одночасна видача команди переміщення механізмів, що можуть рухатись одним шляхом, назустріч один одному.

Повернемось до ситуації нормального відпрацювання кроків технологічного процесу.

Якщо переривання *Int1* відсутнє, за допомогою оператора 15 відбувається перевірка наявність дозволу переходу за адресою наступної

підпрограми – *ST*. У залежності від результату цієї перевірки відбувається розгалуження алгоритму. Або відбувається виконання наступного кроку поточної підпрограми (у такому випадку запускається виконання гілки алгоритму з оператором 25). Або запускається перехід на відпрацювання іншої підпрограми (у такому випадку виконується гілка алгоритму з оператором 26).

Процес формування початкової адреси нової підпрограми забезпечується макрооператором 10 – аналізу станів стохастичних входів. У випадку співпадіння сигналів від стохастичних входів $b_j(t_p)$ з умовами B_{xj}, що записані до матриці $|M_B|$ адреса першого рядка нової підпрограми буде записана до лічильника адреси.

У реальних технологічних процесах зміна координат вектора $a(t_p)$ може відбуватися не тільки у відповідності до запрограмованого алгоритму. Тож постає необхідність безперервного порівняння станів детермінованих входів $a_j(t_p)$ $(j = 1, 2, ..., k)$ з їх очікуваними значеннями A_{ij} $(j = 1, 2, ... , k)$, що містяться в і-му рядку матриці станів $|M_A|$. Виконання процедури порівняння залежить від значення ознаки операції F_p, що перевіряється оператором 8. Якщо ця ознака приймає значення логічної одиниці, то виконується логічна операція «І» (макрооператор 13). Якщо ознака F_p приймає значення логічного нуля, то порівняння відбувається з використанням логіки «АБО» (макрооператор 14). Вибір типу операції залежить від того, яким чином повинен відбуватись перехід до наступного кроку алгоритму – при співпадінні усіх фактичних станів датчиків циклу з їх очікуваними значеннями (логічна операція «І»), або при наявності сигналу хоча б від одного з датчиків (логічна операція «АБО»), спрацювання яких очікується на *p*-му кроці програми.

Також слід вказати, що процес переходу, або до виконання наступного рядка поточної підпрограми, або до виконання іншої підпрограми, може відбуватись не тільки так, як було показано вище, а і також за результатами формування дозволу від внутрішніх програмованих таймерів. За це

відповідають уведені до відповідних гілок алгоритму функціонування вдосконаленого ЛКА ПД оператори 23 і 24.

Додатковий функціонал, що реалізують вбудовані таймери реалізовано за допомогою гілки алгоритму з операторами 7, 11, 12, 20. Оператор 7 призначений для запуску процедури відліку запрограмованого проміжку часу. Оператори 11 і 12 виконують процедуру декрементації значення, що міститься у пам’яті таймера, під час надходження імпульсу на відповідний вивід кристала ПЛІС. За допомогою оператора 20 відбувається процедура переходу до наступного кроку поточної підпрограми (перехід на гілку з оператором 25), або формування адреси переходу до відпрацювання іншої підпрограми (перехід на гілку з оператором 26), це залежить від того який саме блок ЛКА ПД був ініціатором формування відліку часу.

У випадку неспівпадіння сигналів детермінованих входів з їх очікуваними на поточному кроці відпрацювання програми значеннями, макрооператорами 13 або 14 буде сформовано логічний нуль» і на блок індикації буде видано інформацію про таке неспівпадіння стану *i*-го датчика (або ряду датчиків) запрограмованому на даному рядку підпрограми значенню. За цей процес відповідає оператор 22 алгоритму.

Після виконання операторів 25 або 26 відбувається перехід на наступний крок відпрацювання робочої програми (виконується оператор 28) та відбувається перевірка наявності кінця програми (виконується оператор 29). Далі цикл роботи алгоритму повторюється, починаючи з видачі команд керування виконавчими механізмами оператором 3.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено аналіз структурної організації керуючих автоматів з паралельною архітектурою, а саме: математичної моделі ЛКА ПД та структури класичного керуючого автомату паралельної дії.

Досліджено особливості побудови логічних керуючих автоматів паралельної дії із функціями вбудованих програмованих таймерів, у тому числі таких, що задані в категоріях математичної моделі.

Досліджено підходи до паралельного програмування і побудови паралельних алгоритмів, визначено їх переваги та недоліки, обрано ті з них, які дозволяють реалізувати поставлені у роботі задачі. Розглянуто універсальний алгоритм функціонування керуючого автомату паралельної дії, визначено шляхи інтеграції до нього елементів, що реалізують функції програмованих користувачем таймерів.

Виконано розробку алгоритму функціонування керуючого автомату паралельної дії з розширеним функціоналом у вигляді вбудованих програмованих таймерів.

Апробацію результатів дослідження було виконано у збірнику наукових праць «Системи управління навігації та зв’язку» випуск 2 (76) 2024 року [9].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аллашев О. Ю. Спосіб автоматичного синтезу ПЛІС-контролера по технологічній циклограмі / О. Ю.Аллашев, С. Я. Бовчалюк, І. О.Фурман, [та ін.]. Патент на корисну модель №37285, МПК (2006) G05B 19/18. – Опубл. 25.11.2008, Бюл. №22, 2008 р.

2. Аллашев О. Ю. Спосіб підвищення якості керуючих програм для логічних контролерів / О. Ю.Аллашев, С. Я. Бовчалюк, І. О.Фурман, [та ін.]. Патент на корисну модель №39844, МПК (2009) G05B 19/18. – Опубл. 10.03.2009, Бюл. №5, 2009 р.

3. Бовчалюк С. Я. Визначення напрямків розвитку керуючих пристроїв з паралельною архітектурою на базі ПЛІС / С. Я. Бовчалюк, О. М. Піскарьов, С. С. Радченко, [та ін.] // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2023. – Випуск 1 (71). – С. 69-72. ISSN 2073-7394.

4. Бовчалюк С. Я. Концепція реалізації програмних засобів інформаційної технології паралельного логічного керування / С. Я. Бовчалюк // Проблеми енергозабезпечення та енергозбереження в АПК України: Вісник ХНТУСГ імені Петра Василенка, вип. 102 – Харків, 2010. – С. 83–84.

5. Бовчалюк С. Я. Модели, методы и средства информационной технологии параллельного логического управления объектами железнодорожной автоматики: дис. … канд. техн. наук: 05.13.06 / Бовчалюк Станіслав Ярославович. –Харьков, 2008. –203 с.

6. Бовчалюк С. Я. Новая информационная технология логического управления в энергетике и на транспорте / С. Я. Бовчалюк // Системи управління, навігації та зв'язку. – К.: Центральний науково-дослідний інститут навігації і управління, 2007. – Вип. 3 – С. 47-51.

7. Бовчалюк С. Я. HDL-модель програмованого логічного керуючого автомата паралельної дії / С. Я. Бовчалюк, І. О.Фурман // Радіоелектронні і

комп’ютерні системи. – 2007. – №6 (25). – С. 202–205.

8. Деренько М. С. Технічна реалізація промислового зразка ПЛІС-контролера паралельної дії / М. С. Деренько, С. Я. Бовчалюк, И. А.Фурман [та ін.] // Проблеми енергозабезпечення та енергозбереження в АПК України: Вісник ХНТУСГ імені Петра Василенка, вип. 87. – Харків, 2009. – С. 126–127.

9. Коломоєць В. С. Розвиток моделі та структури керуючих пристроїв з паралельною архітектурою / В. С. Коломоєць, Б. М. Коломоєць Я. В. Гаращенко, С. Я. Бовчалюк // Системи управління, навігації та зв’язку. Збірник наукових праць. – Полтава: ПНТУ, 2024. – Випуск 2 (76). – С. 64-66. ISSN 2073-7394.

10. Краснобаев В. А., Фурман И. А., Кошман С. А [та ін.] Концепция, методы и средства моделирования на ПЛИС контроллеров и процессоров с параллельной архитектурой / В. А Краснобаев, И. А. Фурман, С. А. Кошман [та ін.] // Автомобильный транспорт: Сб. научных трудов, вып. 16. – Харьков, 2005. С. 338–341.

11. Краснобаєв В. А. Основні властивості непозиційної системи числення у класі лишків і їх вплив на структуру та принципи реалізації арифметичних операцій комп’ютерної системи / В. А. Краснобаєв, С. О. Кошман, В. М. Курчанов [та ін.] // Системи управління, навігації та зв’язку, 2019. – Вип. 2(54) – С. 114-118.

12. Малиновський М. Л. Програмований логічний контролер / М. Л. Малиновский, С. Я. Бовчалюк, И. А. Фурман. Патент України 71200 А, МПК G05B 19/18. – Опубл. 5.11.2004, Бюл. №11.

13. Малиновський М. Л. Проектирование цифровых устройств на ПЛИС: Учебник для ВУЗов /. Харьков: Факт, 2006. – 164 с.

14. Малиновський М. Л. Развитие архитектуры ПЛК параллельного действия: от абстрактной модели параллельного автомата, до инженерной реализации безопасного ПЛИС-контроллера / С. Я. Бовчалюк, И. А. Фурман, М. Л. Малиновский // Енергетика та комп’ютерно-інтегровані технології в

АПК. 2016. – №. 2 (5). – С. 62-66.

15. Піскарьов О. М. Перспективи побудови інтелектуальних мереж SMART GRID базі ПЛІС-технологій / С. Я. Бовчалюк, С.О. Тимчук, . О. М. Піскарьов [та ін.] // Вісник Вінницького політехнічного інституту. –2017. – №5 (134). – С. 80–85.

16. Тимчук С. О. Безпечний ПЛІС-контролер паралельної дії, як інтелектуальне ядро Smart Grid / С. Я Бовчалюк, С. О. Тимчук, І. О. Фурман [та ін.] // Проблеми енергозабезпечення та енергозбереження в АПК України: Вісник ХНТУСГ імені Петра Василенка, вип. 187. – Харків, 2017. – С. 51–53.

17. Тимчук С. О. Концепція побудови автомата паралельної дії із нечіткою логікою для формування інтелектуального ядра SMART GRID / С. Я. Бовчалюк, С. О.Тимчук // Енергетика та комп'ютерно-інтегровані технології в АПК.– 2017. – № 1(6). – С . 76–79.

18. Фурман И. А. Научно-технические основы создания и промышленного применения параллельных логических контроллеров на программируемых БИС с матричной структурой : дис. … докт. техн. наук: 05.13.05 / Фурман Ілля Олександрович. – К., 1989. – 197 с.

19. Фурман И. А. Перспективы развития структуры и технологии применения параллельных логических контроллеров / И. А. Фурман // Электротехника. – 1990. - №4. – С. 98-100.

20. Фурман И. А. Совершенствование математической модели и архитектуры логических управляющих автоматов параллельного действия / И. А. Фурман, С. Я. Бовчалюк // Інформаційно-керуючі системи на залізничному транспорті. – 2006. – №3(59). – С. 72–76.

21. Фурман И. А. Технологическое визуальное программирование – новое средство автоматизации разработки программного обеспечения ПЛК / И. А.Фурман, С. А. Колесников // Інформаційно-керуючі системи на залізничному транспорті. – 2003. – № 4. – С. 46–48.

22. Фурман І. О Вдосконалення алгоритму функціонування програмованого логічного контролера паралельної дії / І. О. Фурман, С. Я.

Бовчалюк // Інформаційно-керуючі системи на залізничному транспорті. – 2007. – №2 (64). – С. 38–42.

23. Ilya Furman. Development and study of technological visual programming of logic control problems / Ilya Furman, Stanislav Bovchaliuk, Alexander Allashev, Aleksey Piskarev // Eastern-European Journal of Enterprise technologies, – 2017. – № 6/2 (90). –P. 23–31.

24. Stanislav Bovchaliuk. The Architecture of Fuzzy Logic Automat of Parallel Action for the Intelligent Smart Grid Networks / S. Bovchaliuk, S.Tymchuk, S. Shendryk, V. Shendryk // New Technologies, Development and Application III. NT 2020. Lecture Notes in Networks and Systems, vol. 128. Springer, – 2020. – P. 462–468.