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)

Модель реалізації функцій таймерів керуючого автомату паралельної дії

(тема)

Виконав:

студент II курсу, групи СПм-22-4

Гаращенко Я. В.
(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 Гаращенку Ярославу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель реалізації функцій таймерів 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) Математична модель контролерів паралельної дії
3) Структура ППЛК та ПЛК ПД
4) Технічна документація щодо практичної реалізації керуючих структур з паралельною архітектурою

4. Перелік питань, що потрібно опрацювати у роботі
1) Аналіз архітектури ППЛК першого покоління
2) Дослідження структури побудови ПЛІС-контролерів паралельної дії
3) Аналіз типів і принципів функціонування програмованих таймерів сучасних мікроконтролерів
4) Дослідження математичної моделі логічних керуючих автоматів паралельної дії
5) Розробка структури ПЛІС-контролера ПД, з програмованими таймерами
6) Розробка елементів математичної моделі вдосконаленого ПЛІС-контролера ПД
7) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 12 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 архітектури ППЛК першого покоління	02.04.24-09.04.24	
2	Дослідження структури побудови	10.04.24-17.04.24	
	ПЛІС-контролерів паралельної дії		
3	Аналіз програмованих таймерів сучасних	18.04.24-30.04.24	
	мікроконтролерів		
4	Дослідження математичної моделі ЛКА ПД	01.05.24-08.05.24	
5	Розробка структури вдосконаленого	09.05.24-17.05.24	
	ПЛІС-контролера ПД		
6	Розробка елементів математичної моделі	18.05.24-26.05.24	
	вдосконаленого ПЛІС-контролера ПД		
7	Оформлення матеріалів кваліфікаційної роботи	27.05.24-04.06.24	
8	Подання кваліфікаційної роботи керівникові	05.06.24-07.06.24	
	та її попередній захист		
9	Подання кваліфікаційної роботи на	08.06.24-12.06.24	
	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: 75 с., 8 рис., 3 табл., 2 дод., 20 джерел.

КЕРУЮЧІ ПРИСТРОЇ З ПАРАЛЕЛЬНОЮ АРХІТЕКТУРОЮ, МАТЕМАТИЧНА МОДЕЛЬ ЛКА ПД, ПЛІС-КОНТРОЛЕР ПАРАЛЕЛЬНОЇ ДІЇ, СТРУКТУРА ПЛК ПД

Метою кваліфікаційної роботи є вдосконалення математичної моделі і архітектури програмованого логічного контролера паралельної дії на базі ПЛІС, а також розширення його функціональних можливостей, шляхом уведення до структури програмованих таймерів.

У ході виконання кваліфікаційної роботи загальний огляд керуючих структур з паралельною архітектурою – ППЛК першого покоління та ПЛІС-контролера паралельної дії. Розглянуто особливості реалізації вбудованих таймерів у сучасних мікроконтролерах STM та AVR. Розглянуто та досліджено математичну модель логічного керуючого автомату паралельної дії. Розроблено структуру та елементи математичної моделі перспективного ПЛІС-контролера паралельної дії, що включає в себе програмовані користувачем таймери.

ABSTRACT

Master’s thesis: 75 pages, 8 figures, 3 tables, 2 appendices, 20 sources.

CONTROL DEVICES WITH PARALLEL ARCHITECTURE, MATHEMATICAL MODEL OF LKA PA, FPGA-CONTROLLER OF PARALLEL ACTION, STRUCTURE OF PLC PA

The purpose of the qualification work is to improve the mathematical model and architecture of the programmable logic controller of parallel action based on the FPGA, as well as to expand its functionality by introducing programmable timers into the structure.

In the course of the qualification work, a general overview of control structures with parallel architecture – PPLC of the first generation and FPGA-controller of parallel action. Features of implementation of built-in timers in modern STM and AVR microcontrollers are considered. The mathematical model of the logical control automaton of parallel action is considered and researched. The structure and elements of the mathematical model of a promising parallel-FPGA controller, which includes user-programmable timers, have been developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БП – блок пам’яті

ВЛК – вузол логічного керування

ІМС – інтегральна мікросхема

ЛКА – логічний керуючий автомат

ПД – паралельної дії

ПЛІС – програмована логічна інтегральна схема

ПЛК – програмований логічний контролер

ППЛК – паралельний програмований логічний контролер

ПЛК ПД – програмований логічний контролер паралельної дії

ТА – технологічний агрегат

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

ШІМ – широтно-імпульсна модуляція

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

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

ВСТУП

За останнє десятиліття питанням розвитку і вдосконалення інформаційної технології паралельного логічного керування на базі ПЛІС-контролерів паралельної дії було присвячено не дуже багато уваги. У той же час в останніх дослідженнях [2] було показано, що вдосконалення цієї технології, уведення до її складу нового функціоналу, дозволить значно розширити сфери її застосування не тільки для керування об’єктами критичного застосування, але і для побудови систем керування звичайними промисловими об’єктами.

Якщо провести аналіз публікацій, можна визначити декілька основних тенденцій і напрямків розвитку керуючих пристроїв подібних систем, що отримали загальновживану назву – паралельні програмовані логічні контролери (ППЛК), або у сучасній інтерпретації – програмовані логічні контролери паралельної дії (ПЛК ПД):

- розвиток і вдосконалення внутрішньої організації ПЛК ПД [1, 4-7]. Це заходи, що дозволяють покращити безпекові можливостей керуючих систем, що побудовані на їх базі або розширюють можливості інструментів логічної обробки вхідних сигналів. Також тут можна вказати можливості для введення реалізації функцій нечіткого логічного висновку [13, 14, 20];

- реалізація ПЛК ПД на більш сучасній елементній базі – від побудови системи на дискретних ІМС малого ступеня інтеграції [16, 17], до керуючих структур на сучасних кристалах ПЛІС [9, 10];

- вдосконалення мови і технології програмування контролерів паралельної дії – від написання програми на паперовому носії з подальшим ручним програмуванням ІМС ПЗП, до автоматизованої технології програмування TVP (Technological Visual Programming) різних версій [3, 18, 19].

Одним із найбільш перспективних вдосконалень, що значно розширює

функціонал ПЛІС-контролерів паралельної дії є уведення внутрішніх таймерів і лічильників. В одній з останніх публікацій за напрямком дослідження інформаційної технології паралельного логічного керування, а саме – керуючих архітектур паралельної дії, було визначено пріоритетність уведення до таких структур саме внутрішніх програмованих користувачем таймерів і лічильників [2]. У той же час ні математичною моделлю, ні HDL-моделлю, ні алгоритмом функціонування ПЛІС-контролерів паралельної дії (ПД) не передбачено можливості реалізації подібного функціоналу [4-6].

Отже побудова вдосконаленої структури і математичної моделі ПЛК ПД з розширеним функціоналом у вигляді програмованих таймерів, є, на даному етапі досліджень, найбільш пріоритетною задачею.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Загальний огляд керуючих структур з паралельною архітектурою

Одним з ефективних напрямів підвищення якості функціонування сучасних керуючих пристроїв та систем, що використовуються в промисловості і на транспорті, в стаціонарних та мобільних об'єктах, є розпаралелювання процесів обчислень та логічної обробки. Відомі структурні, програмні та інші методи розпаралелювання таких процесів, що реалізуються на контролерах і процесорах з традиційною архітектурою.

1.1.1 ППЛК першого покоління

У роботах [4-7] показано, що суттєве підвищення якості функціонування обчислювальних та керуючих пристроїв і систем може бути досягнуто на основі використання як елементної бази регулярних структур програмованих логічних інтегральних схем (ПЛІС), так спеціальних інструментальних засобів для їх програмування, що дозволяють у достатньо короткі терміни і ціною малих витрат інженерної праці розробляти, моделювати, досліджувати та коригувати архітектури пристроїв та систем з нетрадиційною архітектурою.

В основу методології побудови паралельних керуючих автоматів покладено раціональне поєднання та практичне використання властивостей регулярності, що притаманне об'єктам управління дискретної циклічної дії, технологічним мовам опису алгоритмів управління такими об'єктами, і матричним мікроелектронним структурам [16, 17]. У літературних джерелах, наприклад [10, 16, 17] описані принципи побудови класичного ПЛК паралельної дії (ППЛК) (рисунок 1.1).

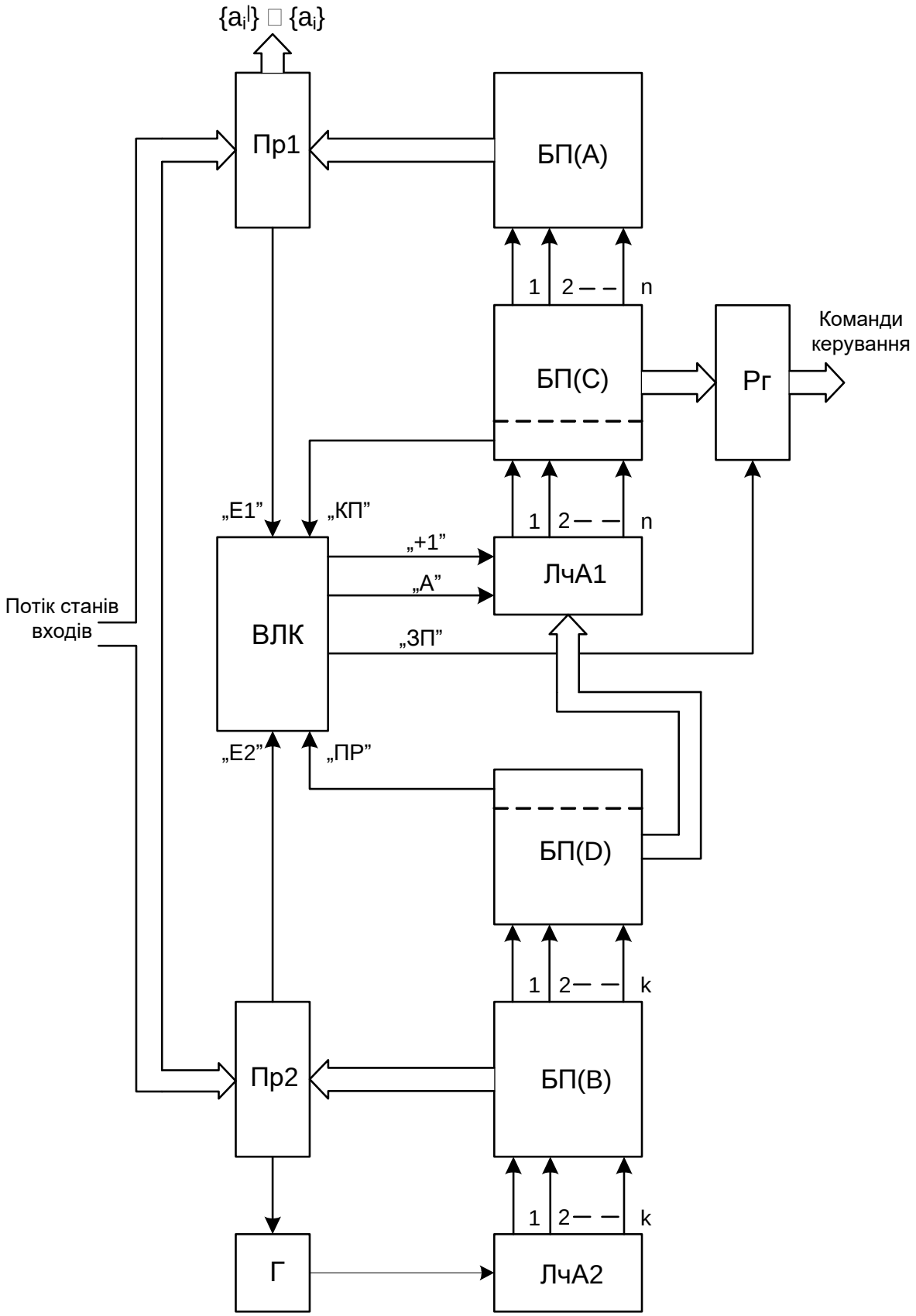

Рисунок 1.1 – Структура класичного ПЛК паралельної дії

Ця структура, що вже стала класичною, була свого часу реалізована на базі дискретних елементів малої і середньої степені інтеграції і показала свою життєздатність у жорстких умовах промислового виробництва. Основу універсального ППЛК складають блоки пам'яті (БП), причому в БП(А) записується матриця А очікуваних станів керованого об'єкта, в БП(В) – матриця В очікуваних станів зовнішнього середовища (ВС), в БП(С) – матриця С керуючих команд, а в БП(D) – матриця D адрес переходів.

Розглянемо принцип дії ППЛК. що полягає у наступному. За допомогою генератора (Г) та лічильника адрес ЛчА2 забезпечується послідовне читання інформації з БП(В), схема порівняння Пр2 при цьому здійснює паралельне логічне порівняння векторів запрограмованого та фактичного станів зовнішнього середовища. Якщо має місце збіг фактичної комбінації з одним із запрограмованих, Пр2 видає у вузол логічного керування (ВЛК) сигнал «Еквівалентність 2» («Е2»), блокує генератор, унаслідок чого ЛчА2 зупиняється, а з БП(D) на адресні входи СчА1 видається код початкової адреси відповідної підпрограми, що записана до блоків пам’яті БП(А) і БП(С). Якщо при цьому попередня підпрограма знаходилася в стадії виконання, перехід до нової підпрограми здійснюється тільки після видачі в кінці виконання попередньої сигналу «Кінець підпрограми» («КП»). Для цього сигналу у всіх рядках матриці С виділено один розряд.

За наявності на вході ВЛК сигналу «КП» (цей сигнал також видається і у початковому стані контролера) ВЛК сигналом «Адреса» («А») забезпечує запис у ЛчА1 початкової адреси обраної підпрограми і, як наслідок,– адресацію відповідного рядка БП(А) та БП(С). У той самий час за сигналом з ВЛК «ЗП» відбувається запис у вихідний реєстр Рг комбінації команд керування виконавчими механізмами.

В результаті виконання команд спрацьовують механізми керованого об’єкту, що призводить до зміни комбінації станів на одному зі входів Пр1, що діє аналогічно Пр2. Ця схема порівняння звіряє комбінацію фактичних

станів керованого об'єкта з тією, що мала бути сформована на даному рядку програми в результаті виконання запрограмованих керуючих команд. При спрацьовуванні Пр1 ВЛК збільшує вміст СчА1 на одиницю, забезпечуючи цим переадресацію БП(А) і БП(С) на наступний рядок підпрограми. Виконання наступних рядків підпрограми здійснюється аналогічно до вже розглянутого алгоритму.

1.1.2 ПЛІС-контролер паралельної дії

На рисунку 1.2 показано базову структуру сучасного ПЛІС-контролера ПД, що покладено в основу промислових зразків і реалізовано на кристалах ПЛІС компанії Altera [5].

Реалізація базових функцій логічного керування відбувається за рахунок перевірки істинності рівнянь (1.1) і (2), що записані у блок логічного керування:

$$A = КП \vee ПР1 \vee ПР2, \tag{1.1}$$

$$+1 = E \wedge \overline{КП} \wedge \overline{ПР1} \wedge \overline{ПР2}, \tag{1.2}$$

де КП – ознака кінця підпрограми;

ПР1 – ознака переривання від БПП;

ПР2 – ознака переривання від БПЗК;

А – початкова адреса підпрограми;

Е – сигнал еквівалентності;

+1 – сигнал переходу до наступного рядка.

Сам керуючий автомат складається із наступних блоків: блоку індикації – БІ; схеми порівняння – СП; блоку вибору операції – БВО; блоку логічного керування – БЛК; лічильника адреси – ЛА; вихідного регістра – ВР; а також блоків пам'яті станів, команд, переходів і заборонених

комбінацій – БПС, БПК, БПП, БПЗК.

Процес відпрацювання керуючої програми складається з двох етапів або частин: 1) аналіз комбінацій станів датчиків умов переходів (станів зовнішнього середовища) і формування початкової адреси підпрограми; 2) власне відпрацювання обраної підпрограми. Причому аналіз станів зовнішнього середовища здійснюється паралельно і незалежно від відпрацювання підпрограми.

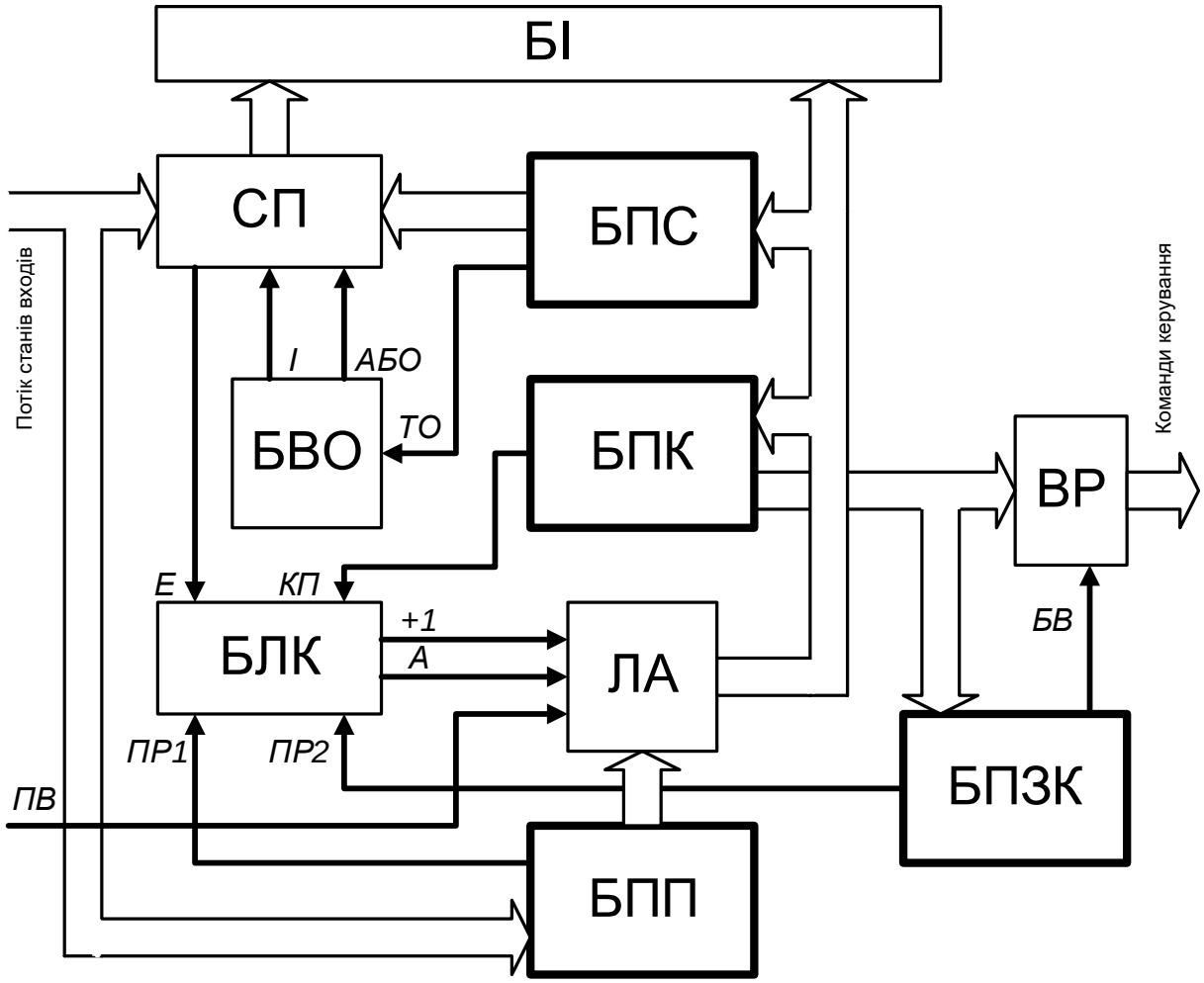

Рисунок 1.2 – Структура ПЛІС-контролера паралельної дії

На показаній структурі за вибір початкової адреси підпрограми відповідає блок пам'яті переходів (БПП), який, у разі появи на його вході однієї із запрограмованих комбінацій, встановлює лічильник адреси у

відповідний даній комбінації стан. Таким чином реалізовано виконання рівняння (1.1), тобто істинною стає рівність *КП=А* і БЛК формує сигнал «*А*» (Адреса), за яким лічильник адреси переадресовує БПС і БПК на перший рядок обраної підпрограми. В останньому рядку кожної підпрограми, а також у нульовому рядку програми записується тільки ознака кінця підпрограми «*КП*», що використовується як дозвіл переходу керуючого автомата до відпрацювання будь-якої із записаних у блоки пам'яті підпрограм.

Власне відпрацювання обраної підпрограми відбувається за рахунок реалізації рівняння (1.2). Якщо рівність виконується, то БЛК формує сигнал «*+1*», за яким ЛА адресує БПС і БПК на наступний (*i+1*) рядок поточної підпрограми. Умовою формування сигналу «*+1*» є поява сигналу «*Е*», який може бути сформований двома способами. Якщо на певному кроці керуючої програми необхідно порівнювати фактичний стан усіх датчиків циклу з їх очікуваними значеннями, то до останнього стовпця і-го рядка, що записаний до блоку пам'яті станів, записується відповідна ознака і блок вибору операції формує сигнал «*І=1*». Цей сигнал перемикає схему порівняння на реалізацію логічної операції «*І*», тобто сигнал еквівалентності *Е* на її виході з'явиться лише у разі збігу усіх фактичних станів датчиків циклу з їх очікуваними значеннями, записаними до і-го рядку БПС. Якщо для переходу на наступний крок підпрограми достатньо спрацювання лише одного датчика, то БВО формує сигнал «*АБО=1*», що перемикає схему порівняння на реалізацію логічної операції «*АБО*».

Ознаки переривань «*ПР1*», «*ПР2*», сигнали «*ПВ*» і «*БВ*», не формують базову логіку роботи автомата, але з технічної точки зору є необхідними. Детально про призначення цих сигналів, а також з більш детальним описом роботи ПЛІС-контролера ПД можна ознайомитись в [4].

1.2 Особливості реалізації вбудованих таймерів у сучасних мікроконтролерах

1.2.1 Таймери у мікроконтролерах STM

Таймер – один з найважливіших для мікроконтролера пристроїв, що застосовується для вирішення безлічі завдань: відлік часу, планування дій, генерація сигналів, вимірювання параметрів сигналів та ін. пристроїв (блоки живлення; конвертери; інвертори з однофазним, трифазним та багатофазним виходом; частотні перетворювачі тощо). Завдяки наявності іншої різноманітної периферії (ADC, DAC, різні інтерфейси тощо), мікроконтролер стає не просто заміною для спеціалізованих мікросхем, орієнтованих на використання у пристроях живлення, а й перевершує їх, забезпечуючи значну гнучкість рахунок можливості програмування. І навіть здатний конкурувати зі спеціалізованими мікросхемами за ціною (за рахунок універсальності, мікроконтролери виробляються великими партіями, тому мають низьку ціну, незважаючи на високу складність).

Розробники пристроїв на базі мікроконтролерів активно використовують таймери, а розробники мікроконтролерів, у свою чергу, намагаються найбільш повно задовольнити потреби користувачів, надаючи їм у розпорядження різні за складністю та можливостями таймери.

Так, типовий представник мікроконтролерів STM32 містить близько десяти таймерів. До них входять як прості таймери з базовим набором функцій, так і таймери дуже складні та багатофункціональні. З одного боку, це дає масу можливостей для професійних розробників, з іншого – перетворює на жах вивчення цього питання для початківців.

У STM32 мікроконтролерів багато таймерів і вони мають дуже багато функцій. Ці таймери відрізняються один від одного схемним рішенням і своїми можливостями, що ще більше ускладнює їхнє вивчення. Значно полегшує життя той факт, що всі таймери в STM мають єдиний принцип

побудови та управління.

Отже таймер – пристрій, який відмірює заданий інтервал часу, після якого виробляється певний сигнал.

Коли таймер є частиною мікроконтролера, способом сигналізації може бути, наприклад, переривання. Зазвичай таймер працює у періодичному режимі, тобто. не зупиняється після формування сигналу, а починає відразу відміряти наступний інтервал такої самої тривалості. В результаті таймер генерує сигнали (переривання) з певною частотою через задані інтервали часу.

Сигнальний режим передбачає те, що таймер відмірює заданий інтервал часу, після закінчення генерує сигнал і зупиняється. Таймер може мати апаратну підтримку сигнального режиму, або цей режим можна легко реалізувати програмно (достатньо вимкнути таймер в обробку переривання або виконати необхідні дії всередині обробника тільки при першому виклику).

Технічно таймер будується з урахуванням лічильника, який підрахує імпульси деякого тактового сигналу. Зазвичай сигнал про спрацювання таймера формується під час переповнення лічильника. Як правило, є можливість керувати модулем перерахунку лічильника таймера (задавати кількість імпульсів між двома послідовними переповненнями лічильника). Це дозволяє за фіксованої частоти тактового сигналу на вході лічильника встановлювати необхідну частоту генерації сигналів від таймера. Крім того, зазвичай є програмний доступ до лічильника таймера, за поточним значенням якого можна з високою роздільною здатністю визначати час, що минув з останнього переповнення лічильника.

Таймер дає можливість мікроконтролеру та програмі вести відлік часу, що визначає два основні напрямки застосування:

– для виконання необхідних дій у задані моменти часу;

– для визначення (фіксації) моментів, коли відбулися деякі цікаві для нас події.

До першого напрямку застосування таймера можна віднести, наприклад, керування зовнішніми пристроями за заздалегідь запланованим розкладом, відповідно до якого в необхідні моменти часу програма посилає сигнали, що управляють, до контрольованих пристроїв. Інтервал часу між двома сусідніми запланованими моментами часу може змінюватись у широких межах – від одиниць періодів тактового сигналу до практично нескінченності (годинник, доба, роки). Незалежно від тривалості інтервалу, його можна встановити з високою точністю і з дуже високою роздільною здатністю в часі – до періоду тактового сигналу.

Здатність таймера відміряти дуже малі інтервали часу відкриває можливість застосування таймера для генерації сигналів: цифрових (для цього достатньо виконувати перемикання стану зовнішнього виведення відповідно до закону зміни сигналу, що синтезується) або аналогових (за рахунок спільної роботи таймера і DAC).

Також таймер можна використовувати для організації середовища виконання програми. Йдеться, перш за все, про реалізацію багатозадачності шляхом перемикання між виконуваними завданнями за сигналом таймера.

Не менш важливим є другий напрямок використання таймера – для фіксації моменту настання подій. Це може бути: ведення журналу подій, вимірювання параметрів зовнішніх сигналів (тривалість імпульсу, період сигналу) тощо. Сюди можна віднести, знову ж таки, розв'язання задач управління, тепер уже у тих випадках, коли реакція на зовнішній сигнал залежить від моменту його надходження.

Види таймерів у STM32.

У мікроконтролерах Cortex-M, до яких належать і мікроконтролери STM32, завжди є принаймні один таймер – системний таймер, наявність якого гарантується вимогами архітектури. Стандартизація системного таймера лише на рівні архітектури значно спрощує його використання. Як мінімум, це забезпечує переносимість коду, який використовує цей таймер між різними мікроконтролерами. Переносність особливо важлива для коду

операційних систем, так що не дивно, що системний таймер зазвичай задіяний для потреб операційної системи, якщо вона використовується, звичайно.

Крім системного таймера, мікроконтролери STM32 мають сторожовий таймер (і навіть не один). Сторожові таймери мають досить вузьку спеціалізацію – вони використовуються для автоматичного перезавантаження у разі зависання програми.

Крім того, мікроконтролери STM32 мають безліч універсальних таймерів, що можна використовувати на власний розсуд. На відміну від системного, вони не є частиною ядра, а периферійними пристроями мікроконтролера. Включають такі типи таймерів: базові, загального призначення і таймери з розширеним управлінням. Базові таймери є найпростішими, відповідно до своєї назви мають лише набір базових функцій. Таймери з розширеним управлінням – найскладніші та мають найбільшу кількість реалізованих функцій.

Для таймерів в мікроконтролерах сімейства STM32 використовується наступна система іменування: назва починається з префікса TIM, за яким слідує номер таймера: TIM1, TIM2 і т.д. Важливо, що номер таймера визначає його тип. І в різних мікроконтролерах таймери з однаковими номерами зазвичай сумісні (однаково влаштовані, мають однаковий набір функцій і управляються однаковим чином за допомогою однакових наборів регістрів; що, однак, не звільняє від необхідності уважного ознайомлення з довідковим керівництвом: не гарантується повна відсутність відмінностей). Такий підхід дуже зручний, оскільки дозволяє легко переходити від одного мікроконтролера до іншого – як у сенсі вивчення, так і при переносі коду. Використовувана в мікроконтролерах STM система іменування таймерів має на увазі те, що нумерація не обов'язково повинна бути послідовною, вона має перепустки, що відповідають відсутнім у конкретній моделі мікроконтролера таймерам. Тому, наприклад, якщо мікроконтролер має таймер з максимальним номером 17 (TIM17), це ще не означає, що всього є 17

універсальних таймерів TIM1. TIM17 – таймери з якимись номерами будуть відсутні.

У якості прикладу розглянемо перелік таймерів, що можуть бути включені в мікроконтролери з лінійки STM32F100xx та вкажемо їхню типову приналежність (таблиця 1.1). Необхідно вказати, що таймери TIM5, TIM12, TIM13, TIM14 можуть бути відсутніми у деяких мікроконтролерах лінійки.

Таблиця 1.1 – Таймери мікроконтролерів лінійки STM32F100xx

Ім’я таймера	Тип таймера
TIM1	Розширений
TIM2	Загального призначення
TIM3	
TIM4	
TIM5*	
TIM6	Базовий
TIM7	
TIM12*	Загального призначення
TIM13*	
TIM14*	
TIM15	
TIM16	
TIM17	

Поряд із нумерацією таймерів, життя розробнику також спрощує те, що всі універсальні таймери в STM32, незважаючи на наявні між ними відмінності, побудовані за єдиним принципом і управляються подібним чином. Більш того, можна виявити подібність навіть між таймерами в STM32 і таймерами в 8-бітних мікроконтролерах сімейства STM8, аж до однакового іменування регістрів управління та бітів у них.

Найбільш повним набором можливостей та функцій має таймер з

розширеним управлінням. Зазвичай мікроконтролер STM містить хоча один такий таймер, він має ім'я TIM1. Деякі моделі можуть мати два або навіть три таймери з розширеним керуванням (TIM1, TIM8, TIM20). Так ось, хитрість у тому, що решта всіх таймерів можна розглядати як спрощені варіанти таймера TIM1, в яких виключені деякі конструктивні елементи і не реалізовані деякі функції. З програмної точки зору робота з таймером здійснюється так само, як з TIM1. Звичайно, з урахуванням того, що деякі біти, що відповідають за нереалізовані функції, переходять до числа зарезервованих. Може бути відсутній цілий регістр, якщо всі його біти відповідають нереалізованим у цьому таймері можливостям. Отже, вивчивши найскладніший таймер TIM1, можна працювати з будь-яким іншим таймером, уточнивши лише доступний йому набір функцій.

Побудова таймера. На рисунку 1.3 зображено спрощену структурну схему таймера в мікроконтролері сімейства STM32. Таймер складається з наступних блоків: модуль, що задає час (Time-base unit); схема керування (Controller); набір каналів таймера (Channels); схема зупинки (схема захисного відключення виходів каналів, Break). Названі блоки тісно взаємодіють, обмінюючись сигналами керування та подій (позначено стрілками на схемі). Таймер підключений до шини мікроконтролера APB, через яку здійснюється доступ до його регістрів, від цієї шини таймер отримує тактовий сигнал, який можна використовувати як внутрішній тактовий сигнал для лічильника таймера (на схемі не показано).

Таймер має можливість взаємодіяти із зовнішнім оточенням, простіше кажучи, має входи та виходи (входи/виходи каналів, вхід ETR, вхід BKIN). Деякі таймери мікроконтролера можуть взаємодіяти між собою: для цього вони мають по одному внутрішньому виходу тригерного сигналу (TRGO) і по 4 внутрішні входи для тригерних сигналів ITR0, ITR1, ITR2, ITR3. Кожен тригерний вхід даного таймера підключений до виходу тригерного іншого певного таймера. А тригерний вихід може бути внутрішньо підключений до деякого тригерного входу одного або кількох таймерів. Режим використання

тригерних входів/виходів таймерів залежить від налаштувань. В таблиці 1.2 показано підключення таймерів мікроконтролерів лінійки STM32F100xx.

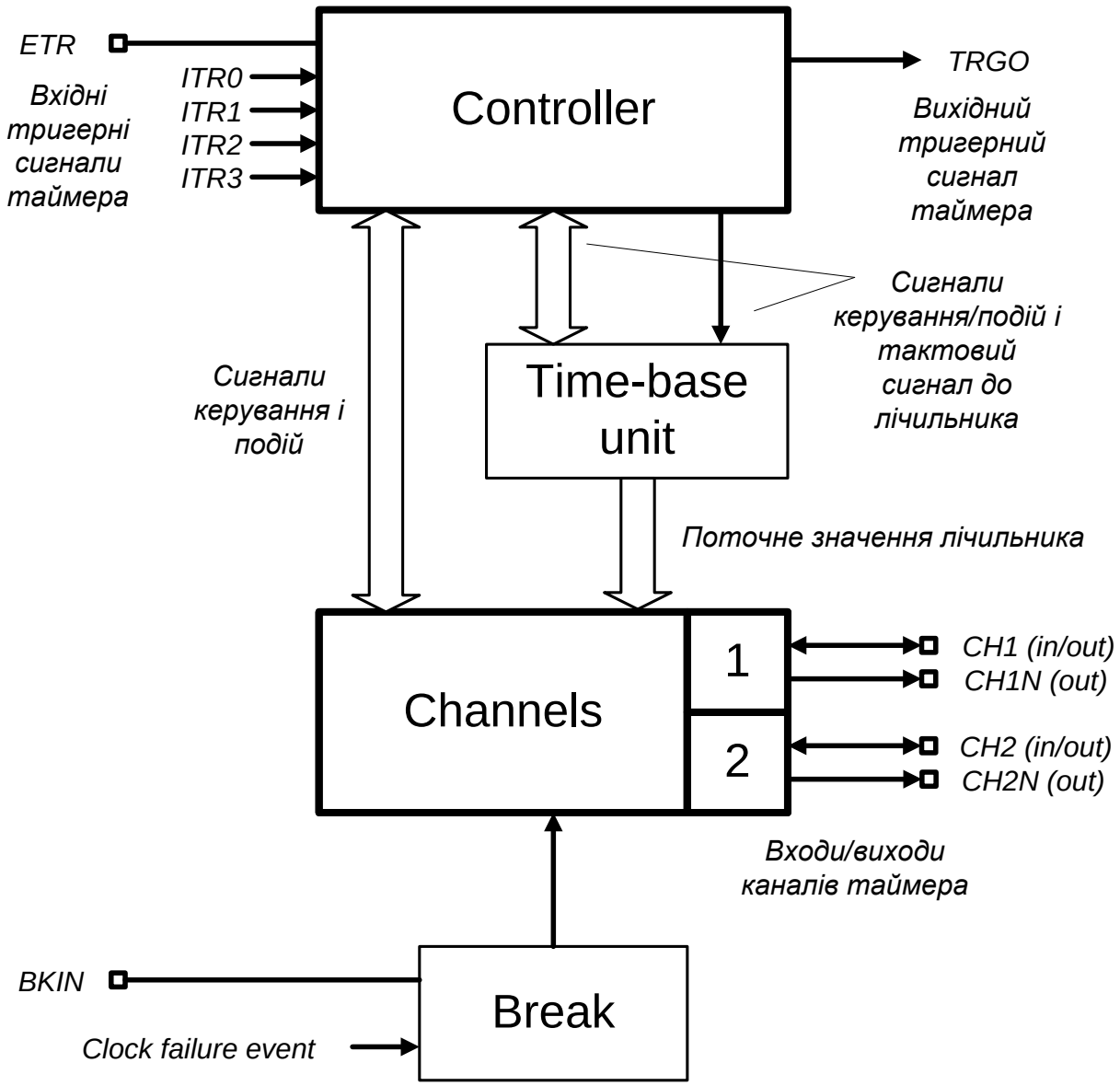

Рисунок 1.3 – Спрощена структурна схема таймера STM32

Основа таймера – модуль, що задає час, головним елементом якого є лічильник. Лічильники універсальних таймерів у мікроконтролерах STM32 є 16-розрядними (за нечисленними винятками: у деяких лінійках мікроконтролерів лічильники TIM2, TIM5 можуть бути 32-розрядними). Лічильник працює спільно з регістром автоперезавантаження, керуючим модулем перерахунку. При рахунку вгору регістр визначає значення, до

якого виконується рахунок, після досягнення якого відбувається переповнення та скидання лічильника. При рахунку вниз регістр визначає початкове значення лічильника, що завантажується під час переповнення. Переповнення формує спеціальний сигнал – подія оновлення, яким виконується ряд важливих дій, і навіть можлива генерація переривання. Тактовий сигнал на лічильник надходить через попередній дільник з керованим коефіцієнтом поділу, який можна змінювати не більше від 1 до 65536 (побудований з урахуванням 16-разрядного лічильника). На вхід попереднього дільника тактовий сигнал надходить зі схеми керування.

Таблиця 1.2 – Підключення таймерів мікроконтролерів лінійки STM32F100xx

Підлеглий таймер	Підключення входів підлеглого до ведучого таймера			
	ITR0	ITR1	ITR2	ITR3
TIM1	TIM15 (TIM5)	TIM2	TIM3	TIM4
TIM2	TIM1	TIM15	TIM3	TIM4
TIM3	TIM1	TIM2	TIM15	TIM4
TIM4	TIM1	TIM2	TIM3	TIM15
TIM12	TIM4	TIM5	TIM13_OC	TIM14_OC
TIM15	TIM2	TIM3	TIM16_OC	TIM17_OC

Схема керування дає можливість вибрати джерело тактового сигналу (це може бути внутрішній тактовий сигнал з шини; зовнішній тригерний сигнал ETR; внутрішній тригерний сигнал; сигнал від енкодер-інтерфейсу). Схема керування також визначає напрямок рахунку (вгору або вниз), якщо в таймері реалізована функція перемикання напрямку рахунку. Напрямок рахунку вибирається за допомогою бітів, що конфігурують, а в деяких режимах залежить від стану таймера і перемикається автоматично. Під час роботи таймера у підпорядкованому режимі схема управління вибирає джерело вхідного тригерного сигналу. Також схема управління генерує вихідний тригерний сигнал, який, залежно від налаштувань, може

формуватися при переповненні лічильника таймера або у відповідь інші події.

Схема керування і модуль, що задає час, – це мінімум того, що входить до складу будь-якого таймера (на малюнку ці блоки виділені кольором). Можливості таймера можуть бути суттєво розширені за рахунок наявності одного або кількох каналів. Канали призначені для виконання найбільш часто зустрічаються на практиці задач з генерації імпульсних сигналів і для вимірювання параметрів зовнішніх імпульсних сигналів і можуть працювати відповідно в режимі виходу або входу. Теоретично те, що виконують канали, можна реалізувати програмно, але апаратна реалізація розвантажує процесор, збільшує швидкодію і точність виконання дій.

У режимі входу канал виконує фіксацію поточного вмісту лічильника таймера зовнішнього сигналу, тобто. зберігає значення у регістрі каналу. Налаштуваннями можна задати джерело сигналу і вибрати подію, за якою відбуватиметься фіксація (по фронту, що наростає, або спаду сигналу). За зафіксованим у регістрі каналу значенню, ми можемо з високою точністю дізнатися, у який час відбулася подія. Те саме можна зробити програмними засобами без використання каналів, а з використанням контролера зовнішніх переривань, налаштувавши його на генерацію переривання у відповідь на зміну сигналу на деякому вході. Тоді, зчитуючи в обробнику цього переривання вміст лічильника таймера, ми визначимо момент, коли відбулася подія, що цікавить нас. Але така реалізація має суттєві недоліки:

- зайве обчислювальне навантаження на процесор, пов'язане з обробкою переривання;

- наявність затримки від моменту настання події до моменту, коли всередині обробника переривання буде прочитано значення лічильника (становить кілька періодів тактового сигналу та призводить до появи систематичної помилки);

- якщо переривання може бути оброблено негайно, величина затримки може зрости на невизначену величину;

- максимальна частота виконання фіксацій виявляється сильно обмежена швидкодією процесора.

У режимі виходу канал використовується для створення сигналів. У регістр каналу записують певне значення. Канал на кожному такті порівнює поточне значення лічильника та значення у своєму регістрі та залежно від результату порівняння (і від налаштувань) змінює стан на своєму виході. Наприклад, за рівності значень може відбуватися формування на виході низького або високого рівня або перемикання рівня на протилежний. Канал може мати не один, а два комплементарні виходи, сигнали на яких змінюються в протифазі, з можливістю гнучко задавати полярністю активних рівнів цих сигналів та час затримки між сигналами під час перемикання. Це дуже зручно при керуванні мостовими та напівмостовими схемами інверторів (затримка або «dead time» вводяться для запобігання наскрізним струмам через керовані ключі моста). Як і у режимі входу, функції під час роботи у режимі виходу можна реалізувати і програмними засобами, але ціною втрати швидкодії та точності.

При використанні таймера для генерації сигналів з ШІМ корисним виявляється блок, призначений для захисного відключення виходів каналу. Це відключення відбувається у разі наявності сигналу на відповідному вході мікроконтролера BKIN або у разі збоїв у системі генерації тактового сигналу мікроконтролера (включення схеми, контрольовані сигнали та їхня полярність визначаються налаштуваннями). Захисне відключення призначене для запобігання пошкодженню керованих силових каскадів у разі аварійних ситуацій. При спрацюванні захисного відключення, виходи каналу перетворюються на певний безпечний стан навіть за відсутності тактового сигналу мікроконтролера. Якщо система не використовується, то при збої в системі тактування, виходи опиняються в статичному, «залиплому» стані. При використанні мікроконтролера, наприклад, у схемі інвертора в системах електроживлення, це призведе до різкого зростання струму через трансформатор та пошкодження елементів схеми. Крім того, зовнішній вхід

може бути підключений до детектора перевищення струму та детектора перенапруги на виході.

Огляд таймерів STM32.

Як вже було згадано раніше, різні таймери в микроконтроллерах STM32 мають різні набори доступних функцій. Для того, щоб було простіше обрати потрібний таймер, розглянемо таблицю 1.3 з їх основними характеристиками.

Таблиця 1.3 – Основні характеристики таймерів мікроконтролерів лінійки STM32F100xx

Таймер	Наявність входу ETK	Наявність Master mode	Наявність Slave mode	Наявність захисного відключення	Направлення рахунку	Наявність лічильника циклів	Кількість каналів	Каналів з компл. виходами	Наявність інтерфейсу енкодера
TIM1	+	+	+	+	+/-	+	4	3	+
TIM2	+	+	+	-	+/-	-	4	0	+
TIM3									
TIM4									
TIM5									
TIM6	-	+	-	-	+	-	0	0	-
TIM7									
TIM12	-	+	+	-	+	-	2	0	-
TIM13	-	-	-	-	+	-	1	0	-
TIM14									
TIM15	-	+	+	+	+	+	2	1	+
TIM16	-	-	-	+	+	+	1	1	-
TIM17									

Усі таймери є 16-розрядними (мають 16-розрядні лічильники), за винятком невеликої кількості моделей мікроконтролерів, у яких TIM2, TIM5 мають 32 розряди. У лінійці STM32F100xx таких пристроїв немає, в інших випадках можна уточнити документацію.

Направлення рахунку для деяких таймерів може бути задано програмно (позначено як «+/-»у таблиці) за допомогою біта DIR у регістрі TIMx_CR1, для решти таймерів рахунок можливий лише вгору.

Тактовим сигналом для таймера може бути тактовий внутрішній сигнал з шини APB, до якої підключений таймер; вихідний тригерний сигнал іншого таймера (під час роботи у підпорядкованому режимі); вхідний сигнал одного з каналів (за наявності таких); сигнал з інтерфейсу енкодера (один із варіантів підпорядкованого режиму). Також деякі таймери мають вхід ETR для зовнішнього тригерного сигналу, який певних конфігураціях може бути тактовим.

Деякі таймери можуть взаємодіяти між собою за допомогою внутрішніх підключень між тригерними виходами одного таймера та тригерними входами інших. Для взаємодії один з таймерів, між якими є з'єднання, налаштовується на роботу в master-режимі, інший – в slave-режимі. Налаштування таймера визначають, у яких випадках формується імпульс вихідного тригерного сигналу (наприклад, при переповненні) і якою буде реакція на тригерний вхідний сигнал. Таймер в той самий час може бути головним по відношенню до якогось таймера і підлеглим по відношенню до іншого.

Крім входу ETR, таймери можуть взаємодіяти із «зовнішнім світом» за допомогою каналів (кількість каналів для кожного таймера вказано в таблиці). При роботі в режимі виходу канал може мати один вихід або два комплементарні виходи, на яких можна отримати два протифазні сигнали, що використовується для генерації PWM сигналів, що управляють мостовими або напівмостовими перетворювачами.

Якщо таймер має хоча б один канал із комплементарними виходами,

він також має схему захисного відключення та лічильник циклів. Захисне відключення призначене для того, щоб у разі аварійної ситуації перевести виходи каналів таймера в безпечний для схеми певний стан, який задається налаштуванням таймера. Залежно від налаштувань схема може спрацьовувати при надходженні зовнішнього сигналу (на вхід BKIN) або при виявленні збоїв у системі тактування мікроконтролера.

Лічильник циклів визначає, як часто оновлюватимуться тіньові регістри таймера, що може бути корисним при генерації PWM сигналів - для того, щоб зробити зміну параметрів PWM сигналу більш плавним.

Таймер TIM1 з розширеним керуванням.

Таймери з розширеним управлінням займають особливе місце серед інших таймерів мікроконтролерів STM. Ці таймери мають найбільш складний пристрій, в них реалізовано найбільшу кількість функцій, що забезпечує максимум можливостей їх використання.

Всі «універсальні» таймери (з розширеним управлінням, загального призначення та базові) мають єдиний принцип побудови та управління: вони складаються з однотипних блоків і управляються за допомогою уніфікованого набору регістрів (точніше, деякого підмножини «максимального» набору; реалізований набір регістрів конкретного таймера залежить від реалізованого у ньому набору функцій). Регістри з однаковим призначенням мають однакове розташування в адресному просторі (у них однакові усунення щодо базової адреси таймера). Те саме стосується і окремих бітів – положення бітів з однаковими функціями в межах регістру однаково. Усі «універсальні» таймери описуються структурою TIM_TypeDef. Все це забезпечує хорошу сумісність таймерів «згори донизу»: складніший таймер може бути задіяний замість більш простого з мінімальною модифікацією коду.

Серед «універсальних» таймерів особливе місце посідає TIM1 – таймер із розширеним керуванням. Він має найбільш складний пристрій та максимальні можливості. Вивчивши даний таймер, можна легко перейти до

вивчення та використання будь-якого іншого таймера – достатньо буде уточнити в документації реалізований у конкретному таймері набір функцій/бітів/регістрів (це буде деяке підмножина такого в TIM1).

Хоча таймери різних мікроконтролерів STM32 і навіть різних сімейств мають високу сумісність, невеликі відмінності можливі. Для того щоб розмова була більш предметною, будемо розглядати пристрій та можливості таймера TIM1 у мікроконтролерах з лінійки STM32F100xx. Там де деталі реалізації можуть відрізнятися в залежності від виконання мікроконтролера, будемо мати на увазі STM32F100RB (якщо ще точніше, то STM32F100RBT6B – мікроконтролер у 64-вивідному LQFP корпусі).

TIM1 може використовуватися для вирішення безлічі різноманітних завдань, включаючи вимірювання тривалості імпульсів вхідного сигналу (input capture), генерацію вихідних сигналів (output compare – формування сигналу схемою порівняння; PWM – генерація сигналу з широтно-імпульсною модуляцією; complementary PWM – комплементарний ШІ паузи).

Мінімальні значення тривалості генерованих імпульсів і період сигналу можуть становити близько десятків наносекунд. Максимальні величини практично необмежені, що забезпечується запобіжником самого таймера і запобіжниками RCC (Reset and clock control – система, що управляє скиданням і тактуванням периферійних пристроїв мікроконтролера), а також можливістю каскадного включення таймерів, коли один таймер керується вихідним сигналом іншого таймера.

Таймер TIM1 та інші таймери загального призначення TIMx є повністю незалежними і не мають жодних загальних ресурсів, що розділяються; кожен таймер має власний набір регістрів. З іншого боку, є механізми здійснення взаємодії таймерів.

Основні можливості таймера TIM1

Розглянемо деякі основні можливості таймера TIM1.

Реалізовано на основі 16-бітового лічильника з автоматичним перезавантаженням початкового значення та з керованим напрямом рахунку

(вгору, вниз або по черзі вгору/вниз).

Має 16-бітовий програмований розподільник для тактового сигналу лічильника, що дозволяє здійснювати розподіл частоти на будь-який цілий коефіцієнт у межах 1..65536.

Має до чотирьох незалежних каналів для введення/виведення сигналів з підтримкою кількох режимів роботи:

- вхідна фіксація (Input Capture, тобто запис вмісту лічильника в регістр фіксації по сигналу на вході каналу; використовується для вимірювання періоду, тривалості імпульсів та інших часових характеристик сигналу);

- формування вихідного сигналу під управлінням схеми порівняння (Output Compare); логічний рівень сигналу на виході визначається в результаті зіставлення значень регістра порівняння та лічильника в даний момент;

- генерація сигналу з широтно-імпульсною модуляцією (PWM, тобто ШІМ), з можливістю вибрати вирівнювання по краю або центру імпульсу;

- генерація вихідного сигналу як одиночного імпульсу.

Має канали з комплементарними виходами із програмованим часом запізнення (або захисною паузою, dead-time); ці канали можна використовувати для отримання протифазних сигналів з ШІМ, необхідних для управління ключовими елементами напівмостових і мостових перетворювачах напруги. Наявність захисної паузи, коли на обох виходах формується замикаючий сигнал для ключів, запобігає виникненню наскрізних струмів через ключові елементи напівмостової/мостової схеми.

Має схему синхронізації для управління таймера зовнішніми сигналами та для забезпечення можливості внутрішньої взаємодії кількох таймерів мікроконтролера.

Має лічильник циклів (repetition counter) для можливості відновлення регістрів таймера тільки після того, як відбудеться задана кількість циклів рахунку (переповнень лічильника).

Має вхід зупинки (break input) для перекладу вихідних сигналів каналів

таймера (для каналів, що працюють у режимі виходу) в деякий зумовлений, безпечний для керованого зовнішнього обладнання стан (наприклад, при керуванні мостовим інвертором, це може бути стан, при якому всі ключі мосту закриті).

Забезпечує можливість створення переривання або запиту DMA у відповідь на наступні події:

а) оновлення:

1) при переповненні/антипереповненні лічильника, якщо при цьому лічильник циклів містить нульове значення;

2) при програмній генерації події оновлення установкою біта UG у регістрі TIMx_EGR, якщо у регістрі TIMx_CR1 біти URS=0, UDIS=0;

3) при генерації події оновлення по тригерному події, якщо у регістрі TIMx_CR1 біти URS = 0, UDIS = 0;

б) тригерна (пускова) подія – trigger event: запуск рахунку, зупинка, ініціалізація або рахунок за сигналом від внутрішнього чи зовнішнього джерела;

в) виконання вхідної фіксації (input capture) при роботі каналу в режимі входу;

г) спрацьовування схеми порівняння (output compare) під час роботи каналу як виходу;

д) сигнал на вході зупинки (break input).

Має підтримку інкрементних датчиків положення (incremental encoder) та схемотехніки з датчиками Холла для вирішення задач просторового позиціонування.

Має вхід, що використовується як тригерний (вхід для запуску заданої дії) або поциклового управління струмом (cycle-by-cycle current management).

1.2.2 Таймери-лічильники мікроконтролерів AVR

Будь-який мікроконтролер серії AVR містить декілька вбудованих таймерів. Причому за своїм призначенням їх можна розділити на дві категорії. До першої категорії належать таймери загального призначення. Другу категорію складає сторожовий таймер. Сторожовий таймер призначений для автоматичного перезапуску мікроконтролера у разі зависання його програми. Зависанням називають зациклювання програми внаслідок помилки, допущеної програмістом чи результаті дії зовнішньої перешкоди. Для кожної мікросхеми потрібен лише один сторожовий таймер. У будь-якому мікроконтролері AVR такий таймер є.

Таймери загального призначення використовуються для формування різних інтервалів часу та прямокутних імпульсів заданої частоти. Крім того, вони можуть працювати в режимі лічильника і підраховувати тактові імпульси заданої частоти, вимірюючи таким чином тривалість зовнішніх сигналів, а також, при необхідності, підраховувати кількість будь-яких зовнішніх імпульсів. З цієї причини ці таймери називають: «таймери/лічильники». У мікросхемах AVR використовуються як восьмирозрядні, так і шістнадцятирозрядні таймери/лічильники. Їхня кількість для різних мікроконтролерів змінюється від одного до чотирьох.

Усі таймери позначаються числами від 0 до 3. Наприклад, Timer/Counter0, Timer/Counter1 тощо. Їх найчастіше називають скорочено T0, T1, T2, T3. Таймери T0 і T2 у більшості мікроконтролерів восьмирозрядні. Таймери T1 та T3 – шістнадцятирозрядні. Таймер T0 є у будь-якій мікросхемі AVR. Інші додаються в міру ускладнення моделі.

Кожен восьмирозрядний таймер є одним восьмирозрядним регістром, який для мікроконтролера є регістром вводу/виводу. Цей регістр зберігає поточне значення таймера і називається лічильним регістром. Шістнадцятирозрядні таймери мають шістнадцятирозрядний рахунковий регістр. Кожен лічильний регістр має своє ім'я. Рахунковий регістр

восьмирозрядного таймера називається TCNTx. Де x – це номер таймера. Для таймера T0 регістр називається TCNT0. Для таймера T2 – TCNT2. Шістнадцятирозрядні регістри називаються схожим чином. Відмінність у тому, що кожен шістнадцятирозрядний лічильний регістр для мікроконтролера являє собою два регістри вводу/виводу. Один для зберігання старших бітів числа, а другий для зберігання молодших бітів. До імені регістру старших розрядів додається буква H, а для регістра молодших розрядів додається буква L. Таким чином, рахунковий регістр таймера T1 – це два регістри вводу/виводу: TCNT1H і TCNT1 L. Рахунковий регістр таймера T3 – це два регістри TCNT3 H і TC L.

Мікроконтролер може записати в будь-який рахунковий регістр будь-яке число в будь-який момент часу, а також у будь-який момент прочитати вміст будь-якого рахункового регістра. Коли таймер входить у режим рахунку, то його вхід починають надходити рахункові імпульси. Після приходу кожного такого імпульсу вміст лічильного регістра збільшується на одиницю. Рахунковими імпульсами можуть бути як спеціальні тактові імпульси, вироблені всередині самого мікроконтролера, і зовнішні імпульси, що надходять на спеціальні входи мікросхеми. При переповненні лічильного регістру його вміст обнулюється, і рахунок починається спочатку.

Будь-який таймер жорстко зав'язаний із системою переривань. Викликати переривання може ціла низка подій, пов'язаних з таймером. Наприклад, існує переривання з переповнення таймера, спрацьовування спеціальної схеми збігу. Окремі переривання може викликати сторожовий таймер.

Режими роботи таймерів.

Таймери мікроконтролерів сімейства AVR можуть працювати у кількох режимах. Різні мікроконтролери мають різні набори режимів для своїх таймерів. Для вибору режимів роботи є спеціальні регістри – регістри управління таймерами. Для звичайних таймерів вживається один регістр управління. Для складніших – два регістри. Регістри керування таймером

називаються TCCRx (де x – номер таймера). Наприклад, для таймера T0 використовується один регістр з ім'ям TCCR0. Для керування таймером T1 використовується два регістри: TCCR1A і TCCR1 B. За допомогою регістрів керування здійснюється не тільки вибір відповідного режиму, але й більш тонка настройка таймера. Розглянемо основні режими роботи таймерів та їх опис.

Нормальний режим (Normal).

Це найпростіший режим. У цьому режимі таймер робить підрахунок імпульсів, що приходять на його вхід (від тактового генератора або зовнішнього пристрою) і викликає переривання за переповненням. Цей режим є єдиним режимом роботи для восьмирозрядних таймерів більшості мікроконтролерів сімейства Tiny та частини мікроконтролерів сімейства Mega. Для решти восьмирозрядних і всіх шістнадцятирозрядних таймерів це лише один з можливих режимів.

Режим «Захоплення» (Capture).

Суть цього режиму полягає у збереженні вмісту рахункового регістра таймера у певний момент часу. Запам'ятовування відбувається або сигналу, що надходить через спеціальний вхід мікроконтролера, або сигналу з виходу вбудованого компаратора. Цей режим зручний у разі, коли необхідно виміряти тривалість будь-якого зовнішнього процесу. Наприклад, час, протягом якого напруга на конденсаторі досягне певного значення. У цьому випадку напруга з конденсатора подається на один із входів компаратора, а на другий його вхід подається опорна напруга. Мікроконтролер повинен одночасно запустити два ці процеси (подати напругу на конденсатор і запустити таймер у режимі Capture). Конденсатор почне заряджатися, напруга на ньому при цьому плавно зростатиме. Одночасно лічильник таймера відраховуватиме тактові імпульси заданої частоти. У той момент, коли напруга на конденсаторі зрівняється з опорною напругою, логічний рівень на виході компаратора зміниться протилежний. За цим сигналом поточне значення рахункового регістру запам'ятовується у спеціальному

регістрі захоплення. Ім'я цього регістру ICRx (для таймера T0 це буде ICR0, для T1 - ICR1 і т.д.). Одночасно виробляється запит на переривання.

Використовуючи принцип вимірювання часу зарядки, зручно створювати прості схеми, що працюють із різними аналоговими датчиками (температури, тиску і т.д.). Якщо принцип роботи датчика полягає у зміні його внутрішнього опору, такий датчик можна включити в ланцюг зарядки конденсатора. Ємнісні датчики можна підключати безпосередньо.

Режим «Скидання при збігу» (CTC).

Для роботи в режимі CTC використовується спеціальний регістр – регістр збігу. Якщо мікроконтролер містить кілька таймерів, то для кожного з них існує свій окремий регістр збігу. Причому для восьмирозрядних таймерів регістр збігу – це один восьмирозрядний регістр. Для шістнадцятирозрядних таймерів регістр збігу – це два восьмирозрядні регістри. Регістри порівняння також мають свої імена. Наприклад, регістр збігу таймера T1 складається з двох регістрів: OCR1 L і OCR1 H. У ряді мікроконтролерів існують два регістри збігу. Так у всіх мікроконтролерах сімейства Tiny існує два регістри збігу для таймера T1. Це регістри OCR1 A та OCR1 B. Два регістри збігу для таймера T1 має і мікроконтролер Atmega8 x. У другому випадку як таймер, так і його регістри збігу мають шістнадцять розрядів. Якщо регістр збігу шістнадцятирозрядний, то фізично складається з двох регістрів вводу/вывода. Наприклад, два регістри збігу таймера T1 мікросхеми Atmega8 x є чотири регістри вводу/виводу з іменами OCR1 AL, OCR1 AH, OCR1 BL, OCR1 BH.

Ці регістри включаються лише тоді, коли вибрано режим CTC. У цьому режимі, як і в попередньому, таймер здійснює підрахунок вхідних імпульсів. Поточне значення таймера з його лічильного регістра постійно порівнюється зі вмістом регістрів збігу. Якщо таймер має два регістри збігу, то для кожного з цих регістрів проводиться окреме порівняння. Коли вміст рахункового регістру збігатиметься з вмістом одного з регістрів збігу, буде виклик відповідного переривання. Окрім виклику переривання в момент

збігу може відбуватися одна з наступних подій:

– скидання таймера (правильно тільки для регістрів збігу OCR1 та OCR1 A);

– зміна стану одного з висновків мікроконтролера (правильно для всіх регістрів).

Відбудеться або не станеться одна або обидві, з перерахованих вище подій, визначається налаштуваннями таймера.

Режим «Швидкодіючий ШІМ» (Fast PWM).

Сигнал із ШІМ (англійською «Pulse Width Modulation» – PWM) часто використовується у пристроях управління. Сигнал із ШІМ можна, наприклад, використовувати для регулювання швидкості обертання електродвигуна постійного струму. Для цього замість постійної напруги на двигун подається прямокутна імпульсна напруга. Завдяки інерції двигуна імпульси згладжуються, і двигун обертається рівномірно. Змінюючи скважність імпульсів (тобто відношення періоду імпульсів до їх тривалості), можна змінювати середню напругу, що прикладена до двигуна і тим самим змінювати швидкість його обертання. Так само можна керувати й іншими пристроями, наприклад, нагрівальними елементами, освітлювальними приладами тощо. Перевага імпульсного управління у високому ККД. Імпульсні керуючі елементи розсіюють набагато менше паразитної потужності, ніж керуючі елементи, що працюють у аналоговому режимі.

Для формування сигналу ШІМ використовуються ті самі регістри збігу, що працюють і в режимі CTC. Робота таймера у режимі Fast PWM показано на рисунку 1.4. Сигнал із ШІМ формується на спеціальному виході мікроконтролера. На вхід таймера подаються імпульси системного генератора. Таймер перебуває у стані безперервного рахунку. При переповненні таймера вміст скидається в нуль, і рахунок починається спочатку. У режимі ШІМ переповнення таймера не викликає переривань. На рисунку 1.4 це показано у вигляді пилкоподібної кривої позначеної як TCNTn. Крива є залежністю вмісту рахункового регістру від часу.

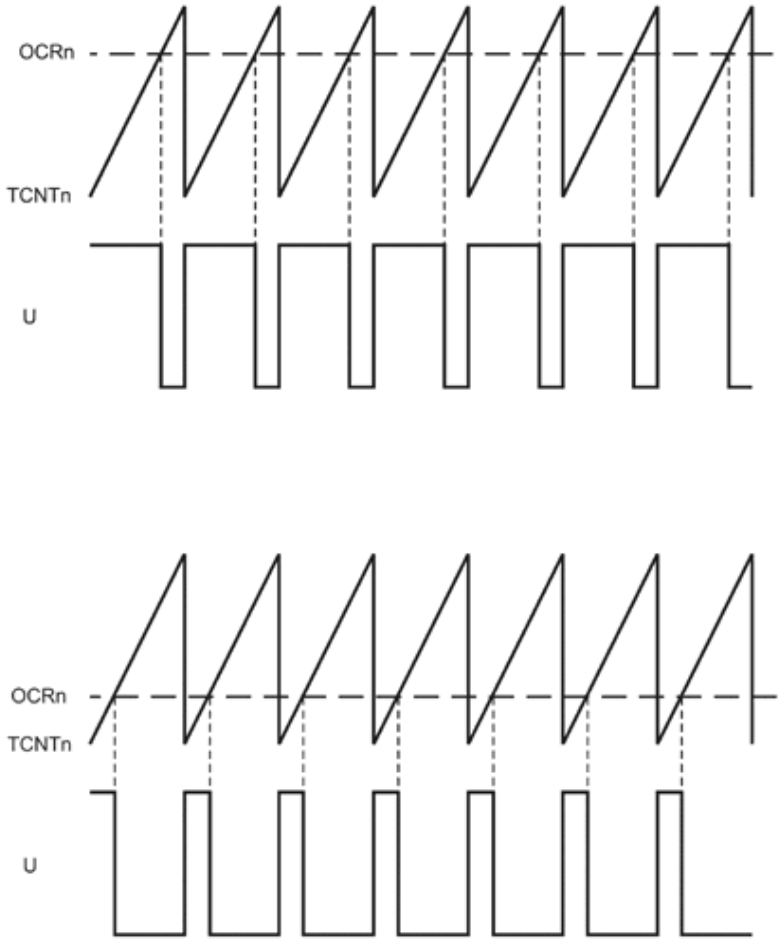

Рисунок 1.4 – Робота таймера у режимі Fast PWM

Вміст лічильного регістру безперервно порівнюється зі вмістом регістру збігу. Поки число в регістрі OCRn більше, ніж число в лічильному регістрі таймера (TCNTn), напруга на виході ШІМ дорівнює логічній одиниці. Коли ж у процесі рахунку вміст рахункового регістра TCNTn стане більшим за вміст OCRn, на виході ШІМ встановиться нульовий потенціал. В результаті на виході ми отримаємо прямокутні імпульси. Добре ці імпульси будуть залежати від вмісту регістра OCRn. Чим менше число OCRn, тим вище скважність вихідних імпульсів. Якщо вміст OCRn досягне свого максимального значення, то імпульси на виході ШІМ зникнуть, і там постійно буде присутня логічна одиниця. При зменшенні числа OCRn з'являться імпульси малої скважності (тривалість майже дорівнює періоду).

Якщо плавно зменшувати число OCRn, то скважність буде плавно зменшуватися. Коли вміст OCRn досягне нуля імпульси на виході ШІМ, також зникнуть і там встановиться логічний нуль.

Режим «ШИМ із фазовою корекцією» (Phase Correct PWM).

Попередньо розглянутий режим ШІМ Fast PWM має один недолік. При зміні тривалості імпульсів змінюється і його фаза. За керування електродвигуном така поведінка фази не бажана. Тому в мікроконтролерах AVR передбачено ще один режим ШІМ. Це ШІМ із точною фазою. Принцип роботи таймера у цьому режимі зображено рисунку 1.5.

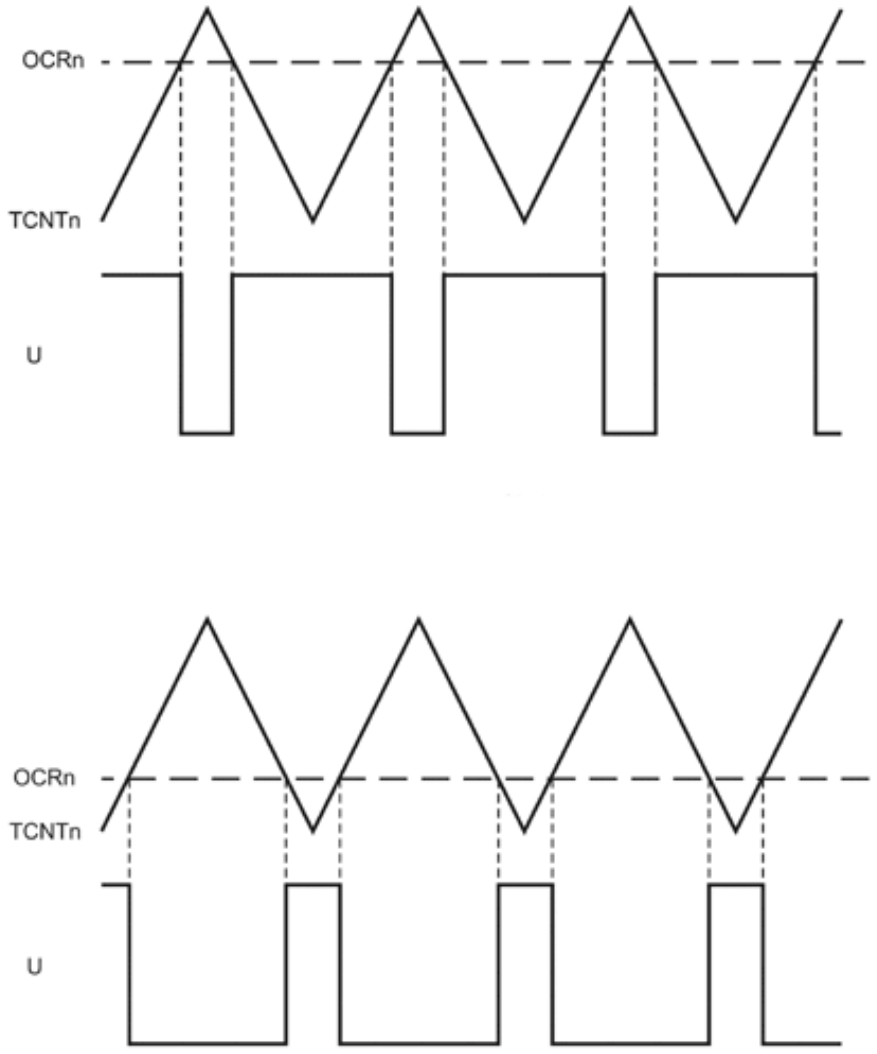

Рисунок 1.5 – Робота таймера у режимі Phase Correct PWM

Відмінність режиму «Phase Correct PWM» від режиму «Fast PWM» в режимі роботи лічильника. Спочатку лічильник рахує так само, як і в попередньому режимі (від кожного вхідного імпульсу його значення збільшується на одиницю). Досягши свого максимального значення, лічильник не скидається в нуль, а перемикається в режим реверсивного рахунку. Тепер від кожного вхідного імпульсу його вміст зменшується на одиницю. В результаті, пилкоподібна крива, що відображає вміст лічильного регістра TCNTn стає симетричною. Система збігу робота так само, як і в попередньому випадку. Завдяки симетричності сигналу на таймері фаза вихідних імпульсів у процесі регулювання скважності не змінюється. Середина кожного імпульсу точно прив’язана до точки зміни напряму рахунку таймера.

Недоліком режиму Phase Correct PWM можна вважати вдвічі меншу частоту вихідного сигналу. Це суттєво зменшує динамічність регулювання. Крім того, при використанні зовнішніх фільтрів, для перетворення імпульсного сигналу ШІМ в аналоговий, схема з більш низькою частотою вимагатиме застосування комплектуючих з великими габаритами та масою.

Асинхронний режим.

У деяких моделях мікроконтролерів таймер може працювати в асинхронному режимі. У цьому режимі на вхід таймера подається частота від внутрішнього кварцового генератора, або від зовнішнього генератора. Лічильник не виробляє жодних переривань та додаткових сигналів. У цьому режимі він працює як годинник реального часу. Мікроконтролер може встановлювати вміст рахункового регістру. А потім у будь-який момент він може рахувати цей вміст, отримавши таким чином поточне значення реального часу.

Попередні дільники таймерів/лічильників.

Кожен таймер мікроконтролера може працювати від двох різних джерел тактових імпульсів. Або це зовнішні імпульси, або імпульси, що виробляються внутрішньою схемою мікроконтролера. Яке джерело сигналів

не було обрано, перед тим, як потрапити на вхід таймера, цей сигнал проходить схему попереднього дільника. Попередній дільник призначений для того, щоб розширити діапазон формованих частот і тривалостей таймера. Кожна мікросхема AVR має власну структуру попереднього дільника для таймерів/лічильників. Спрощена схема одного з варіантів попереднього дільника наведена на рисунку 1.6.

Як видно із схеми, частота внутрішнього тактового генератора CLK надходить на спеціальний десятирозрядний дільник. З виходів дільника знімається сигнали, CLK/8, CLK/32, CLK/64, CLK/128, CLK/256 та CLK/1024. Всі ці сигнали надходять на входи даних мультиплексора. На адресні входи мультиплексора надходять сигнали від трьох розрядів регістру керування таймером (TCCRn). Таким чином, записуючи в розряди CSn0, CSn1, CSn2 різні значення можна вибирати один з восьми режимів роботи ділільника. Залежно від обраного режиму вихід схеми можуть надходити сигнал із однієї з виходів десятиразрядного дільника, прямий сигнал з тактового генератора чи нульовий логічний рівень (входу D0). В останньому випадку сигнал на вході таймера відсутній і його робота припиняється.

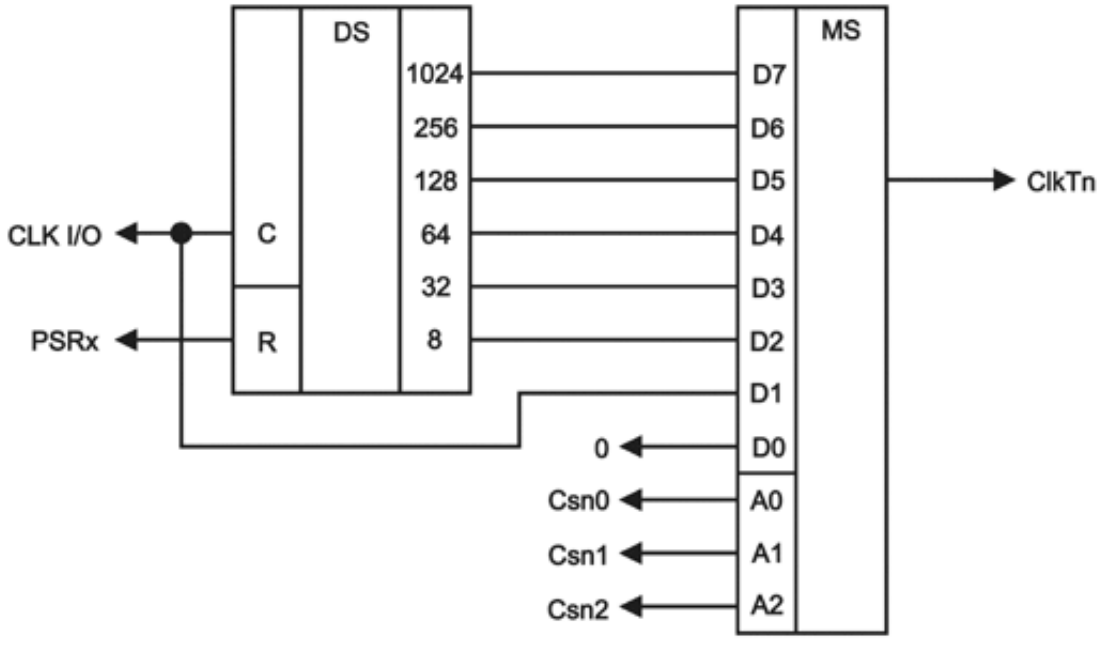

Рисунок 1.6 – Попередній ділільник мікроконтролера AVR

Схема, що наведена на рисунку 1.6, не є стандартом для всіх мікроконтролерів серії AVR. Вона відбиває лише загальний принцип побудови ділільників. У різних моделях це зроблено по різному. На рисунку 1.7 наведено ще одну схему. Ця схема, на відміну попередньої, передбачає подачу на входи таймерів, тактового сигналу від зовнішнього джерела. Для цього кількість сигналів, що знімаються з десятирозрядного дільника, зменшено до чотирьох. CLK/32 та CLK/128 виключені. Натомість у схемі з'явилися зв'язки, якими на вхід таймера може надходити зовнішні імпульси. Ці імпульси мають подаватися на вхід Tn. З цього входу імпульси надходять на формувач, який здійснює їхню попередню обробку (наближає їхню форму до прямокутної). Потім імпульси надходять на вхід D7 дешифратора. На вхід D6 надходять самі імпульси, але тільки в інвертованому вигляді.

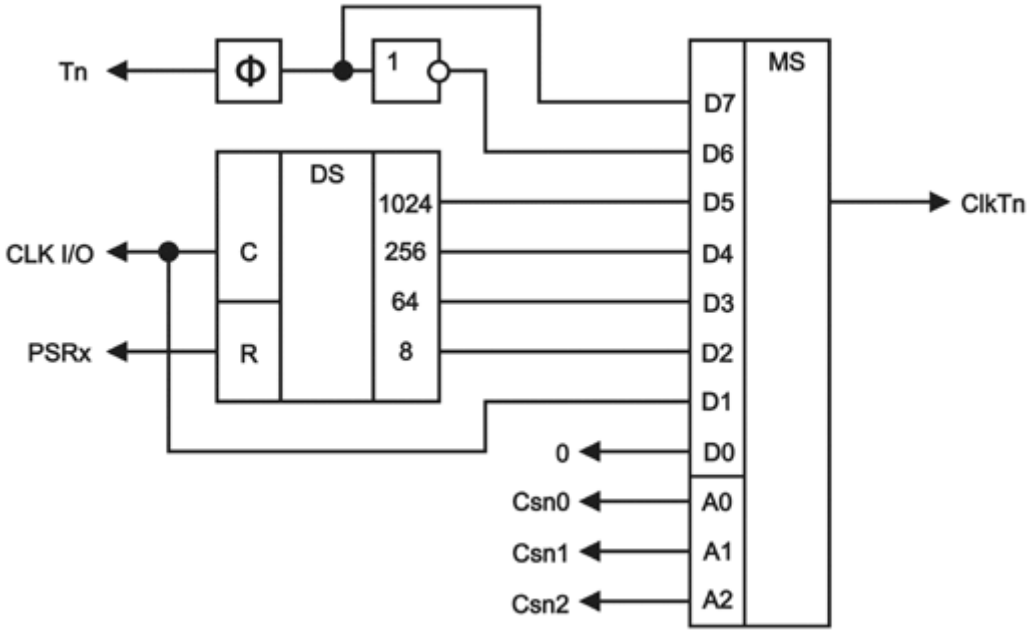

Рисунок 1.7 – Попередній ділільник з входом зовнішнього сигналу

У результаті для схеми, що показана на рисунку 1.7, ми отримуємо наступні вісім режимів роботи:

– режим 0 – відсутність імпульсів;

– режим 1 – прямий сигнал від внутрішнього генератора;

– режими 2-5 – один із сигналів з дільника;

– режим 6 – інверсний сигнал із зовнішнього входу;

– режим 7 – прямий зовнішній сигнал.

1.3 Постановка завдання дослідження

Попередній аналіз показав, що керуючі пристрої з паралельною архітектурою, як було показано в [4-7], мають низку переваг. Серед таких переваг необхідно згадати дві найважливіші. Першою з них є практично повна відсутність залежності кількості контрольованих входів і керованих виходів від швидкодії контролера. Другою перевагою є можливість реалізації TVP-технології для автоматизованого створення керуючих програм неспеціалістом в області програмування і за допомогою спрощених мов. Така технологія дозволяє значно зменшити кількість помилок у програмному коді, пришвидшити процес його формування, 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 дуже актуальною задачею.

Враховуючи вищевказане, основною метою роботи є вдосконалення математичної моделі і архітектури програмованого логічного контролера паралельної дії на базі ПЛІС, а також розширення його функціональних можливостей, шляхом уведення до структури програмованих таймерів.

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

- провести аналіз керуючих структур з паралельною архітектурою, розглянути їх переваги та недоліки, визначити шляхи вдосконалення;

- провести дослідження особливостей реалізації вбудованих таймерів у сучасних мікроконтролерах;

- розглянути існуючі математичні моделі керуючих автоматів з паралельною архітектурою, визначити шляхи імплементації до існуючих моделей додаткових елементів;

- розробити математичну модель і архітектуру ПЛІС-контролера паралельної дії, що містять у собі реалізацію програмованих таймерів.

2 МАТЕМАТИЧНА МОДЕЛЬ ЛКА ПД

Для чіткого розуміння різниці між керуючими автоматами послідовної і паралельної дії, необхідно розглянути принцип, за яким їх можна чітко диференціювати.

В загальному розумінні, як послідовні, так і паралельні програмовані логічні контролери можна характеризувати тривалістю циклу однократного обслуговування входів, що контролюються і виходів керування:

$$T_ц = t_I + t_{II}, \tag{2.1}$$

де $T_ц$ – тривалість циклу однократного обслуговування всіх входів-виходів контролера в одиницях дискретного автоматного часу;

t_I – дискретний автоматний час, що витрачається на аналіз станів входів;

t_{II} – дискретний автоматний час, що витрачається на формування команд керування.

Послідовними ПЛК будемо називати контролери, в яких використовується послідовний принцип обслуговування контрольованих входів і керованих виходів. Для таких контролерів дискретний автоматний час можна визначити, як:

$$t^I = \sum_{i=1}^{k} t_i^I, \qquad t^{II} = \sum_{j=1}^{m} t_j^{II}, \tag{2.2}$$

де k – кількість контрольованих входів контролера;

m –кількість команд керування виконавчими механізмами.

Паралельними ПЛК прийнято називати контролери, в яких

використовується паралельний принцип обслуговування входів-виходів. Для таких контролерів час визначається наступними виразами:

$$t^{I}=t_i^{I}(i=1, 2, \dots, k), \qquad t^{II}=t_j^{II}(j=1, 2, \dots, m). \tag{2.3}$$

Враховуючи вищевказану диференціацію, розглянемо математичну модель класичного ЛКА ПД, що на практиці реалізує паралельний принцип обробки інформації і паралельний принцип формування команд керування керованим об’єктом [4].

В основу побудови математичної моделі функціонування паралельних автоматів і ПЛК, покладено наступне узагальнене формалізоване описання поведінки технологічних агрегатів дискретної циклічної дії ($ТА_{ДЦ}$).

$ТА_{ДЦ}$ – це об'єкти, поведінка яких у часі та просторі строго детермінована і може бути описана наступними параметрами. Кінцевою кількістю станів, у яких протягом циклу роботи агрегату знаходяться механізми:

$$C=\{c_1, c_2, \dots, c_m\} \tag{2.4}$$

Кінцевою кількістю станів датчиків, що контролюють стани відповідних механізмів:

$$A=\{a_1, a_2, \dots, a_k\}. \tag{2.5}$$

Таким чином цикл роботи $ТА_{ДЦ}$ можна описати кінцевою кількістю інтервалів дискретного автоматного часу:

$$T=\{t_1, t_2, \dots, t_s\}, \tag{2.6}$$

Зауважемо. що для кожного *i*-го інтервалу існує кінцева підмножина

(комбінація) станів датчиків:

$$A_i=\{a_{i1}, a_{i2}, \dots , a_{ik}\}, A_i \subset A, \tag{2.7}$$

Вищевказана множина є єдиною, що «дозволена» для вмикання відповідної єдиної кінцевої підмножини (або комбінації) механізмів:

$$C_i=\{c_{i1}, c_{i2}, \dots ,c_{im}\}\ C_i \subset C. \tag{2.8}$$

Отже, циклограму роботи $ТА_{ДЦ}$ можна задати двома прямокутними матрицями кінцевих розмірів: матрицею станів датчиків *A* і матрицею станів механізмів *C*. Причьому вектори-рядки у них розміщені строго детерміновано і кожному *i*-му рядку матриці *A* однозначно відповідає *i*-й рядок матриці *C*:

$$A=\begin{Vmatrix} a_{11}\ a_{12}\ \dots a_{1k} \\ a_{21}\ a_{22}\ \dots a_{2k} \\ \dots\ \dots\ \dots\ \dots \\ a_{s1}\ a_{s2}\ \dots a_{sk} \end{Vmatrix}; \qquad C=\begin{Vmatrix} c_{11}\ c_{12}\ \dots c_{1m} \\ c_{21}\ c_{22}\ \dots c_{2m} \\ \dots\ \dots\ \dots\ \dots \\ c_{s1}\ c_{s2}\ \dots c_{sm} \end{Vmatrix}. \tag{2.9}$$

В матриці *A* кількість стовпців *k* відповідає кількості контрольованих датчиків, встановлених на механізмах технологічного агрегату. А кількість стовпців *m* в матриці *C* відповідає числу механізмів керування агрегатом. Кількість рядків *s* в обох матрицях є однаковою і дорівнює числу рядків (або іншими словами – кроків, етапів) циклограми.

Можна стверджувати, що керування технологічним обладнанням дискретної дії переважно здійснюється за значеннями кінцевих станів механізмів і датчиків (тобто значеннями «включено» чи «виключено»). Таким чином циклограма роботи такого устаткування може бути описана за допомогою булевих матриць. Отже, у якості математичної моделі

функціонування ТА$_{ДЦ}$ можна використати булеві матриці типу C_{im} і A_{ik} з детермінованим розміщенням у них векторів-рядків станів механізмів і векторів-рядків станів відповідних датчиків.

Взаємодія наступних векторів дає змогу виконати формльний опис керуючого автомату паралельної дії:

- вектор *a* – вектор станів. Це комбінація станів органів керування і датчиків керованого технологічного агрегату, значення яких є детермінованими, тобто чітко визначеними у часі. Довжина вектора дорівнює *k*;

- вектор *в*. Вектор умов – комбінація станів зовнішнього середовища. У загальному випадку цє є умови розгалуження алгоритму керування, тобто вони не можуть бути визначені заздалегідь, а саме є стохастичними. Довжина вектора – *u*;

- вектор *c* – вектор керування. Комбінація сигналів, яка видається на виконавчі механізми ТА. Довжина вектора – *m*.

Також розглянемо додаткові вектори, що описують внутрішні стани керуючого автомату:

- вектор *d* – вектор адреси переходу. Цей вектор являє собою номер рядка матриць *A* і *C* (тобто фактично являє собою адресу) початку підпрограми в алгоритмі з розгалуженнями. Довжина вектора – *q*;

- вектор *e* – вектор заборонених станів. Додатковий вектор удосконаленої структури ПЛК паралельної дії. Це комбінація команд керування виконавчими механізмами, що є забороненою, тобто такою, поява якої на виході керуючого автомата може призвести до аварійного ситуації на керованому об’єкті.

Множина векторів a_i утворює матрицю *A* – станів, розміром $s \times к$. Множина векторів c_i формує матрицю *C* – команд, розміром $s \times m$. Вказані матриці призначені для зберігання очікуваних станів датчиків і команд керування ТА. Слід окремо зауважити, що кожному із векторів *a* і *c* поточних значень станів і керувань у кожний момент дискретного автоматного часу

однозначно ставиться у відповідність конкретний рядок з матриць *A* і *C*.

Ще одна матриця математичної моделі – матриця *B* адрес переходів, розміром $q \times u$. Матриця складається з рядків, кожний з яких є умовою переходу до підпрограми. Кількість рядків визначається алгоритмом функціонування технологічного агрегату У загальному випадку, у випадку відсутності розгалужень, матриця *B* може бути відсутня. Кількість стовпців матриці однозначно визначається розміром вектора *в*. Необхідно також зазначити, що у процесі роботи ЛКА відбувається відображення рядка матриці *B* у вектор *d*, при цьому розміри векторів *в* і *d* у загальному випадку не співпадають, тобто $u \neq log_2(s)$.

У випадку опису удосконаленої структури ЛКА ПД. що відзначається покращеними показниками надійності роботи керованого об’єкта за рахунок унеможливлення видачі керуючим автоматом комбінації вихідних сигналів, що можуть призвести до аварії, використовується матриця *E* – заборонених станів. Число рядків і стовпців матриці визначається алгоритмом функціонування керованого об’єкта і в загальному випадку дорівнює $r \times m$, де *m* – кількість керуючих сигналів, а *r* – кількість заборонених їх комбінацій.

Власне алгоритм взаємодії розглянутих вище векторів, визначає принцип функціонування ЛКА ПД і може бути представлений наступним чином.

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 *C* і зберігає їх до початку наступного, *p+1*-го кроку алгоритму:

$$c_j(t_p) = C_{ij}, \tag{2.10}$$

У данному випадку індекс *j* приймає значення від 1 до *m*, а номер рядка визначається на *(p-1)*-ому кроці алгоритму:

$$I = g(p\text{-}1), \tag{2.11}$$

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

Для того, щоб унеможливити формування комбінації вихідних сигналів, які можуть призвести до аварії, значення вектора $c_j(t_p)$ перевіряються на співпадіння зі значенням векторів e_{xj} $(x=1, 2, ..., r; j=1, 2, ..., m)$,тобто значеннями що зберігаються у матриці E:

$$\varepsilon_p = \bigvee_{x=1}^{r} [\bigwedge_{j=1}^{m} (c_{ij} = e_{xj})] \tag{2.12}$$

.

Якщо значення булевої функції ε приймає значення одиниці, це означає, що на виході ЛКА ПД сформувалась заборонена комбінація, Це призводить До блокування видачі цих сигналів на керований об’єкт, айого робота припиняється до моменту усунення умов, що породили цю ситуацію.

Якщо функція ε приймає значення логічного нуля, то керуючі впливи призводять до зміни стану керованого об’єкта, який контролюється системою відповідних датчиків. У математичній моделі така зміна має своє відображення у зміні координат вектора станів $a(t_p)$. Оскільки зміна координат вектора $a(t_p)$ у реальному об'єкті керування може відбуватися не тільки у відповідності до запрограмованого алгоритму, то необхідно безперервно порівнювати стани детермінованих входів $a_j(t_p)$ $(j=1, 2, ..., k)$ з їх очікуваними значеннями A_{ij} $(j=1, 2, ... , k)$, що записані до i-го рядку матриці станів A. У процесі виконання такого порівняння у залежності від значення ознаки F_p виконується логічна одна з двох операції – «І» чи «АБО». Це залежності від того, яким чином повинен відбуватись перехід до наступного кроку алгоритму – або при співпадінні усіх фактичних станів датчиків циклу з їх очікуваними значеннями (у такому випадку виконується порівняння за логічною операцією «І»), або при наявності сигналу хоча б від одного з

датчиків (випадок виконання логічної операції «АБО»), спрацювання яких очікується на *p*-му кроці програми:

$$\alpha_p = \begin{cases} \bigwedge_{j=1}^{k} (a_j(t_p) = A_{ij}), \text{ при } F_p = 0 \\ \bigvee_{j=1}^{k} (a_j(t_p) = A_{ij}), \text{ при } F_p = 1 \end{cases}. \quad (2.13)$$

Якщо функція $\alpha_p=0$, то номер рядка матриці програми не змінюється $I=g(p-1)$. Якщо ж булева функція α_p приймає значення логічної одиниці, то

$$I = g(p) = g(p-1)+1, \text{ якщо } (ST_p \vee Int1_p) = 0, \quad (2.14)$$

де ST_p – ознака дозволу переходу;

$Int1_p$ – ознака безумовного переходу.

Ознака *ST* – дозвіл переходу, може нести в собі різний сенс або різне логічне навантаження. Це може бути кінець відпрацювання попередньої підпрограми, якщо це алгоритми із розгалуженнями, або кінець відпрацювання тіла циклу, якщо це алгоритм з циклами. Якщо ST_p=0, це означає, що на *p*-му кроці виконується поточна підпрограма і необхідно виконати перехід на її наступний крок. Якщо $ST_p=1$ перехід відбувається за адресою переходу, що визначається станом стохастичних входів $b_j(t_p)$. При цьому також необхідно враховувати наявність переривання $Int1_p$. Це перривання вказує на те, що на *p*-му кроці алгоритму виникла ситуація при якій необхідно терміново почати відпрацювання іншої підпрограми (наприклад це може бути підпрограма аварійної зупинки обладнання). Необхідно також вказати, що процес відпрацювання алгоритму у загальному випадку полягає у послідовному виконанні рядків програми. Лише у місцях розгалуження, номер рядка може збільшуватись не на одиницю, тобто може

прийняти будь яке значення адреси переходу, які визначаються наявними сигналами стохастичних входів.

Для формування адреси переходу у програмах з розгалуженнями необхідно порівнювати стан стохастичних входів $b_j(t_p)$ з умовами B_{xj}, що записані до матриці B. У результаті формується булева функція β_p:

$$\beta_p = \bigvee_{x=1}^{n} [\bigwedge_{j=1}^{u} (b_j(t_p) = B_{xj})], \tag{2.15}$$

У даній функції індекси j і x пробігають значення від 1 до u і n відповідно. Тобто виконується аналіз усієї матриці B одночасно. У випадку формування значення булевої функції β_p =1 виконується визначення того або іншого рядка програми для нового $(p+1)$-го кроку алгоритму за виразом:

$$I = g(p) = [d_p = f(b_p)], \text{ якщо } (b_p \wedge ST_p) \vee Int1_p = 1, \tag{2.16}$$

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

На цьому етапі обов’язково враховується значення додаткової умови ST_p та наявність переривання $Int1_p$. $ST_p=0$. Вони означають, що на p-му кроці виконується попередня підпрограма, тобто перехід до наступної є неможливм. Таким чином крок алгоритму визначається тільки станом детермінованих входів $a_j(t_p)$, а отже необхідно дочекатись закінчення відпрацювання попередньої підпрограми і появи значення ознаки $ST_p=1$. Якщо на p-му кроці алгоритму виникає ситуація, при якій необхідно терміново почати відпрацювання іншої підпрограми, то значення переривання $Int1_p$ дорівнює одиниці і перехід відбувається без урахування значення ST_p.

При переході до наступного $(p+1)$-го кроку алгоритму керуючі впливи і очікувані стани визначаються $g(p)$-м рядком матриць A, C і, відповідно, цикл

повторюється аналогічно до розглянутого вище алгоритму.

Аналіз розглянутої математичної моделі показує, що на її базі можна побудувати вдосконалений ЛКА ПД, що реалізуватиме функції програмованих таймерів. Для цього до неї потрібно внести певні зміни, а саме – додати вектори внутрішніх змінних. та функції їхної взаємодії.

3 РОЗРОБКА СТРУКТУРИ І МАТЕМАТИЧНОЇ МОДЕЛІ ПЕРСПЕКТИВНОГО ПЛІС-КОНТРОЛЕРА ПД

3.1 Розробка структури ПЛІС-контролера ПД з програмованими таймерами

Розгляд структури і математичної моделі керуючого автомату паралельної дії, що був проведений вище, показує, що, якщо підходити до реалізації функцій програмованих таймерів в таких автоматах, то необхідно визначитись із тим, чи будуть внутрішні змінні таймерів (і які саме) приймати участь в обох рівняннях (1.1) і (1.2), або лише в одному з них. Аналіз цього питання призводить до висновку, що необхідно повноцінно реалізовувати функцію таймерів, як для реалізації переходу на наступний рядок поточної підпрограми, так і для переходу до іншої підпрограми, тобто мають бути модифіковані обидва рівняння,що описують роботу структури автомату. При цьому спрацювання таймера може безпосередньо ініціювати перехід на наступний рядок (тобто таймер може безпосередньо виконувати формування сигналу «*+1*»). Але сформувати адресу переходу до іншої підпрограми таймер не може, оскільки адреса формується тільки завдяки виконанню процедури аналізу станів стохастичних входів, тобто станів датчиків зовнішнього середовища. Але з іншого боку сигнал від внутрішніх таймерів може приймати участь у формуванні сигналу «*КП*», або іншого еквівалентного за своєю суттю сигналу,який може бути інтерпретованим як участь внутрішніх таймерів у формуванні процедури переходу до іншої підпрограми.

Таким чином для реалізації функцій програмованих таймерів необхідно внести зміни до елементів структури ЛКА ПД (рисунок 1.2), що приймають участь як у формуванні сигналів «*+1*», так і наприклад сигналу «*КП*».

Отже, з урахуванням вищевказаного, пропонується наступна структура

ЛКА ПД (або ПЛІС-контролера паралельної дії, якщо говорити у контексті функціонально завершеного пристрою) у якій реалізовано функції внутрішніх таймерів, показана на рисунку 3.1.

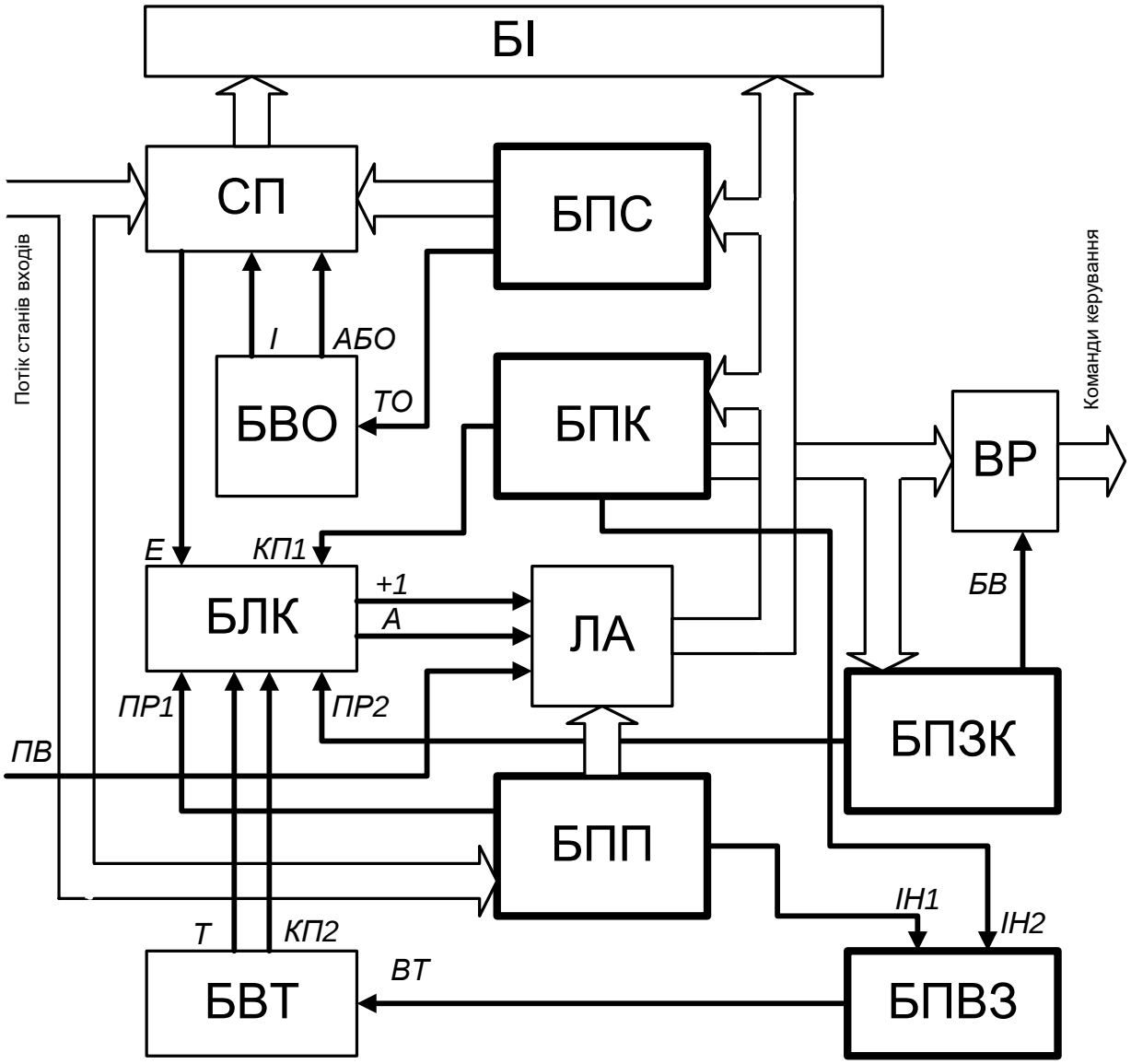

Рисунок 3.1 – Структура ПЛІС-контролера ПД з елементами реалізації внутрішніх програмованих таймерів

Розглянемо основні відмінності і особливості функціонування пропонованої структури.

Базовою відмінністю вдосконаленої структури є наявність ще одного додаткового блоку пам’яті – блоку пам’яті внутрішніх змінних (БПВЗ).

Додатковий блок пам’яті призначений для зберігання чисельних значень внутрішніх таймерів. Також до структури включено ще один додатковий функціональний блок – блок внутрішніх таймерів (БВТ), Саме БВТ дозволяє інтегрувати логіку роботи з внутрішніми таймерами до загальної логіки роботи ПЛК ПД. Насправді БВТ може бути об’єднано з БЛК, але у такому випадку значно ускладниться опис і загальне розуміння роботи ЛКА ПД у цілому.

Послідовно розглянемо призначення додаткових сигналів і блоків, що з’явились на структурі (рисунок 3.1).

Сигнали «*ІН1*» та «*ІН2*» є сигналами ініціаторами запуску відліку внутрішніх таймерів, значення яких записано в *і*-му рядку блоку пам’яті команд (БПК), або блоку пам’яті переходів (БПП). Слід зазначити. що власне процедура відліку і незалежною від усіх інших процесів, що протікають в ЛКА ПД і виконується паралельною з іншими процесами, що протікають в автоматі. Реалізація процедури відліку виконується незалежними апаратними засобами кристалу ПЛІС. Тобто виконання процедури відліку проміжку часу внутрішнього таймера ніяк не залежить від внутрішньої логіки роботи ЛКА ПД, що була розглянута у першому розділі роботи. У момент часу, що відповідає моменту закінчення формування інтервалу відліку таймера БПВЗ формує сигнал «*ВТ*». Цей сигнал фізично означає, що внутрішній таймер закінчив свою роботу і потрібний проміжок часу сформовано. За значенням сигналу «*ВТ*» БВТ ініціює формування або сигналу «*Т*», або «*КП2*» у залежності від того, який саме блок (БПК, або БПП) був ініціатором формування внутрішнього відліку часу.

За наявністю сигналу «*Т*», БЛК виконує формування сигналу «*+1*», тобто ініціює перехід до наступного *і+1*-го рядка поточної підпрограми. Таким чином можна зробити висновок про те. що логіка роботи з внутрішнім таймером фактично відповідає логіці обробки сигналів детермінованих входів. Але є і певні відмінності, що полягають у тому, що при роботі з детермінованими входами схема порівняння очікує появи єдиної

«дозволеної» комбінації вхідних сигналів, а при роботі з таймером «єдина дозволена» комбінація сформується «автоматично» через певний проміжок часу, що записаний до БПВЗ.

Таким чином формування сигналу «*+1*» відбуватиметься у відповідності до наступного формалізованого представлення:

$$+1 = (E \vee T) \wedge \overline{КП1} \wedge \overline{КП2} \wedge \overline{ПР1} \wedge \overline{ПР2} \tag{3.1}$$

На наступному етапі розглянемо логіку роботи автомату при переході до іншої підпрограми. Якщо ініціатором відліку був БПП, то БВТ ініціює формування сигналу «*КП2*», логіка роботи з яким повністю відповідає логіці роботи з сигналом «*КП*», якій у пропонованій структурі отримує іншу назву – «*КП1*». Таким чином перехід до іншої підпрограми буде можливий лише по закінченню сформованого внутрішнім таймером відліку часу. Отже такий алгоритм роботи з підпрограмами повністю аналогічний до ситуації, коли ознака кінця підпрограми фіксується БПК у звичайних умовах (без участі внутрішніх програмованих таймерів).

Таким чином формування сигналу «*A*» у пропонованій структурі відбувається у відповідності до наступного рівняння:

$$A = КП1 \vee КП2 \vee ПР1 \vee ПР2 \tag{3.2}$$

3.2 Розробка елементів математичної моделі вдосконаленого ПЛІС-контролера ПД

Розглянемо зміни і особливості математичної моделі ЛКА ПД, що була розглянута у розділі 2 роботи. Це дозволить описати реалізований додатковий функціонал у вигляді програмованих таймерів.

Враховуючи, що базова логіка роботи ЛКА ПД залишається

незмінною, на початку визначимо ті складові математичної моделі, що не змінили свого функціоналу (або змінили несуттєво).

Залишається абсолютно незмінною базова складова ЛКА ПД у вигляді представлення циклограми роботи $ТА_{ДЦ}$ двома прямокутними матрицями кінцевих розмірів: матрицею станів датчиків A і матрицею станів механізмів C (3.1), що складаються з підмножина станів датчиків A_i (3.2) та кінцевої підмножини механізмів C_i (3.3):

$$A = \begin{Vmatrix} a_{11}\ a_{12}\ \dots a_{1k} \\ a_{21}\ a_{22}\ \dots a_{2k} \\ \dots\ \dots\ \dots\ \dots \\ a_{s1}\ a_{s2}\ \dots a_{sk} \end{Vmatrix}; \qquad C = \begin{Vmatrix} c_{11}\ c_{12}\ \dots c_{1m} \\ c_{21}\ c_{22}\ \dots c_{2m} \\ \dots\ \dots\ \dots\ \dots \\ c_{s1}\ c_{s2}\ \dots c_{sm} \end{Vmatrix}. \tag{3.1}$$

$$A_i=\{a_{i1}, a_{i2}, \dots, a_{ik}\}, A_i \subset A, \tag{3.2}$$

$$C_i=\{c_{i1}, c_{i2}, \dots, c_{im}\}\ C_i \subset C. \tag{3.3}$$

Також не претерпіла ніяких змін заємодія векторів *a*, *в* і *c* станів, умов і керування. Вектори *d* і *e* адреси переходу і заборонених станів також присутні та не змінили свого призначення.

Для відображення у математичній моделі БПВЗ введемо додатковий вектор *iv (internal variable)* – вектор внутрішніх змінних, що у контексті реалізації функцій програмованих таймерів, являє собою чисельне значення проміжку часу, що формується таймером. У даному випадку розглядається ситуація реалізації найпростішого режиму роботи таймеру (наприклад режим Normal мікроконтролерів AVR). Сукупність векторів *iv* формує матрицю *VR* – внутрішніх змінних. Число рядків і стовпців матриці визначається (у найпростішому випадку) кількістю і розрядністю таймерів, і в загальному випадку дорівнює $w \times v$, де w – кількість програмованих таймерів, а v – розрядність відповідних таймерів.

Розглянемо реалізацію процесу переходу на наступний крок поточної підпрограми, або перходу до іншої підпрограми у контексті наявності в ЛКА ПД таймера.

Аналогічно до розглянутої у розділі 2 процедури роботи автомату, почнемо розгляд з p-го кроку алгоритму функціонування керованого об'єкта у проміжку між моментами часу t_p і t_{p+1}. Процес видачі вектора керувань $C(t_p)$, перевірки наявності заборонених комбінацій і відповідності координат вектора $a(t_p)$ значенням A_{ij} $(j=1, 2, \dots, k)$, що записані до i-го рядку матриці станів A, залишається незмінною. Також для формування адреси переходу у програмах з розгалуженнями аналогічно порівнюється стан стохастичних входів $b_j(t_p)$ з умовами B_{xj}, що записані до матриці B.

Введемо додаткову змінну vr_p , що приймає значення логічного нуля, якщо відповідний таймер деактивовано, або він завершив відлік запрограмованого часу. Якщо ж функція $vr_p=1$ – це означає, що таймер виконує процедуру формування запрограмованого проміжку часу. Таким чином якщо $vr_p=0$ робота пропонованого ЛКА відбувається аналогічно до роботи класичної структури ЛКА ПД. У випадку коли функція $vr_p=1$, робота керуючого автомата фактично «затримується» до моменту завершення формування запрограмованого інтервалу часу.

Таким чином для переходу на наступний рядок поточної підпрограми у відповідності до рівняння (3.4) необхідно виконання не тільки умови коли булева функція α_p приймає значення логічної одиниці, але й, додатково $vr_p=0$.

$$I=g(p)=g(p-1)+1, \text{ якщо } (ST_p \wedge Int1_p \wedge \alpha_p \wedge vr_p)=0. \qquad (3.4)$$

Так само для переходу до іншої підпрограми у відповідності до умови (3.5), необхідно щоб окрім формування значення булевої функції β_p =1, також дозвіл від таймера також було сформовано.

$$I=g(p)=[dp=f(bp)], \text{ якщо } [(b_p \wedge ST_p) \vee Int1_p=1] \wedge (\alpha_p \wedge vr_p)=0 \, . \qquad (3.5)$$

Розглянута математична модель не є повною, оскільки не розглядаються процедури запуску таймерів, а також різні варіанти роботи таймерів. Ці моменти ще вимагають подальших досліджень. Але у відповідності до пропонованої структури та елементів математичної моделі можна приступати до моделювання роботи удосконаленого ПЛК ПД на базі сучасних кристалів ПЛІС.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено загальний огляд керуючих структур з паралельною архітектурою, а саме розглянуто ПЛК ПД першого покоління, що будувались на дискретних ІМС; та ПЛІС-контролери ПД, що побудовані на базі сучасних чипів – ПЛІС. Розглянуто особливості реалізації вбудованих таймерів у сучасних мікроконтролерах STM та AVR, показано принципову можливість інтеграції до структури ПЛК ПД таймерів, програмованих користувачем.

Розглянуто математичну модель ПЛК ПД, визначено можливості і підходи до імплементації програмованих таймерів у такоу модель.

Виконано розробку структури вдосконаленого програмованого логічного контролера паралельної дії, у складі якого присутні елементі, що реалізують додатковий функціонал у вигляді таймера, програмованого користувачем.

Вдосконалено математичну модель ЛКА ПД, що містить у собі елементи реалізації таймерів та опис їх функціонування.

Результати наукових досліджень кваліфікаційної роботи опубліковано у збірнику наукових праць «Системи управління навігації та зв’язку» випуск 2 (76) 2024 року [8].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бовчалюк С. Я. Вдосконалення алгоритму функціонування програмованого логічного контролера паралельної дії / С. Я. Бовчалюк, І. О. Фурман // Інформаційно-керуючі системи на залізничному транспорті. – 2007. – №2 (64). – С. 38–42.

2. 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.

3. 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.

4. 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.

5. 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.

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

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

9. 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.

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 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.

13. Піскаорьов 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.

14. 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.

15. 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.

16. 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.

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

18. 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.

19. 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.

20. 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.