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)

HDL-модель керуючого автомату паралельної дії

(тема)

Виконав:

студент II курсу, групи СПм-22-4

Коломоєць Б. М.
(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 Борису Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи HDL-модель 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 ПЛІС-контролера паралельної дії
3) HDL-модель класичного ПЛІС-контролера паралельної дії
4) Технічна документація щодо практичної реалізації керуючих структур з паралельною архітектурою

4. Перелік питань, що потрібно опрацювати у роботі
1) Аналіз структури побудови класичного ПЛІС-контролерів паралельної дії
2) Аналіз особливостей функціонування ПЛІС-контролерів ПД із розширеними функціональними можливостями
3) Аналіз ПЛІС, як перспективного засобу побудови керуючих пристроїв паралельною архітектурою
4) Дослідження HDL-моделі класичного ПЛІС-контролера паралельної дії
5) Дослідження інструментальних засобів розробки HDL-моделей керуючих пристроїв з паралельною архітектурою
6) Розробка HDL-моделі ЛКА ПД з вбудованими програмованими таймерами
7) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 13 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 структури побудови класичного	02.04.24-08.04.24	
	ПЛІС-контролерів паралельної дії		
2	Аналіз особливостей функціонування	09.04.24-115.04.24	
	вдосконаленого ПЛІС-контролера		
3	Дослідження ПЛІС, як бази для HDL-моделі	17.04.24-25.04.24	
4	Дослідження HDL-моделі класичного	26.04.24-03.05.24	
	ПЛІС-контролерів паралельної дії		
5	Дослідження інструментальних засобів	04.05.24-15.05.24	
	розробки HDL-моделей		
6	Розробка HDL-моделі ЛКА ПД з вбудованими	16.05.24-26.05.24	
	програмованими таймерами		
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: 72 с., 17 рис., 1 табл., 2 дод., 20 джерел.

КЕРУЮЧІ ПРИСТРОЇ З ПАРАЛЕЛЬНОЮ АРХІТЕКТУРОЮ, ПЛІС-КОНТРОЛЕР ПАРАЛЕЛЬНОЇ ДІЇ, САПР, СТРУКТУРА ПЛК ПД, HDL-МОДЕЛЬ

Метою кваліфікаційної роботи є розробка HDL-моделі логічного керуючого автомату паралельної дії з розширеним функціоналом у вигляді програмованих таймерів, для формування можливості його практичної реалізації на базі ПЛІС.

У ході виконання кваліфікаційної роботи проведено аналіз керуючих пристроїв з паралельною архітектурою – класичного ПЛІС-контролера паралельної дії та ПЛІС-контролера паралельної дії з розширеними функціональними можливостями. Розглянуто перспективний засіб реалізації керуючих пристроїв з паралельною архітектурою – програмовані логічні інтегральні схеми. Виконано аналіз HDL-моделі класичного ПЛІС-контролера паралельної дії. Досліджено інструментальні засоби розробки HDL-моделей керуючих пристроїв з паралельною архітектурою. Виконано розробку HDL-моделі перспективного ПЛІС-контролера ПД з програмованими таймерами.

ABSTRACT

Master’s thesis: 72 pages, 17 figures, 1 tables, 2 appendices, 20 sources.

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

The goal of the qualification work is to develop an HDL-model of a parallel action logic controller with extended functionality in the form of programmable timers, to form the possibility of its practical implementation on the basis of FPGA.

In the course of the qualification work, an analysis of control devices with parallel architecture was carried out – a classic CPLD-controller of parallel action and a CPLD- controller of parallel action with extended functionality. A promising means of implementing control devices with a parallel architecture – programmable logic integrated circuits – is considered. The analysis of the HDL-model of the classical CPLD-controller of parallel action was performed. Instrumental means of developing HDL-models of control devices with parallel architecture were studied. The development of the structure of a promising CPLD-controller PA with programmable timers was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БП – блок пам’яті

ВІС – велика інтегральна схема

ВЛК – вузол логічного керування

ІМС – інтегральна мікросхема

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

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

У роботах [1, 3, 4, 7] показано можливості щодо суттєвого підвищення якості функціонування керуючих та обчислювальних систем і пристроїв, яке може бути досягнуто на основі використання у якості елементної бази регулярних мікроелектронних структур, а саме – програмованих логічних інтегральних схем (ПЛІС). Також суттєвий вклад у підвищення якості може внести використання спеціальних інструментальних засобів для конфігурування і програмування ПЛІС – спеціалізованих САПР, що дозволяють у достатньо короткі терміни і ціною малих витрат інженерної праці розробляти, моделювати, досліджувати та коригувати архітектури пристроїв та систем з нетрадиційною архітектурою.

За останні роки питанням дослідження і розвитку інформаційної технології паралельного логічного керування на базі ПЛІС-контролерів ПД присвячувалось не дуже багато уваги. Але у тих наукових роботах, де ці питання розглядались, наприклад [11-13] було показано, що розвиток цієї технології має достатньо широкі перспективи. Уведення до ПЛІС-контролерів паралельної дії, як ядра такої технології, додаткових функцій, дозволятиме значно розширити сфери її застосування. Таким чином з’явиться можливість більш широкого її застосування не тільки для керування об’єктами критичної дії, але і для побудови систем керування класичними промисловими об’єктами і технологічними агрегатами.

Аналіз публікацій за темою наукового дослідження показав, що існує декілька основних напрямків і тенденцій розвитку керуючих пристроїв з паралельною архітектурою на базі ПЛІС. Ось ці основні напрямки і тенденції:

- побудова програмованих логічних контролерів паралельної дії на елементній базі більшої ступені інтеграції від дискретних мікросхем до надвеликих ВІС, а саме сучасних кристалів програмованих логічних

інтегральних схем [3. 8, 14, 15];

- розвиток і модернізація внутрішньої структури контролерів паралельної дії. Серед прикладів такого розвитку можна вказати розширення безпекових спроможностей керуючих систем, що побудовані на їх базі, за рахунок унеможливлення видачі на керований об’єкт заборонених комбінацій команд керування; розширення інструментарію логічної обробки вхідних сигналів стохастичних входів. за рахунок уведення можливості їх порівняння за логікою не тільки «І». але й «АБО»; уведення елементів що дозволяють реалізувати функції нечіткого логічного висновку [3, 4, 7, 12, 13];

- вдосконалення мови і, що є значно більш важливим – вдосконалення технології програмування керуючих пристроїв з паралельною архітектурою. У якості конкретних прикладів можна згадати поступову трансформацію технології формування керуючої програми від її написання на паперовому носії мовою ЯППЛК з подальшим ручним програмуванням ІМС ПЗП, до більш розвиненої мови ЯПЛК-М, яка дозволяла формувати керуючу програму із застосуванням ЕОМ [2, 19]. Також у цьому контексті слід згадати формування 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 [17, 19], 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.

На думку автора, одним із найбільш перспективних вдосконалень, що дозволяє значно розширити функціонал ПЛІС-контролерів паралельної дії є можливість уведення до їх структури внутрішніх програмованих користувачем таймерів і лічильників. В одній з останніх публікацій за напрямком розвитку і модернізації інформаційної технології паралельного логічного керування, було показано, що саме уведення програмованих таймерів є актуальною і пріоритетною задачею на даному етапі. У

колективній роботі [6] було запропоновано вдосконалену архітектуру ЛКА ПД, що містить у своєму складі елементи, що реалізують функціонал програмованих таймерів. У той самий час, обґрунтування вдосконаленої структури з боку математичної моделі або HDL-моделі наведено не було.

Таким чином побудова HDL-моделі ПЛК ПД з розширеними функціональними можливостями у вигляді програмованих користувачем таймерів, є, на даному етапі досліджень, однією з найбільш пріоритетних задач.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Керуючі структури з паралельною архітектурою

В основу методології побудови керуючих автоматів із паралельною архітектурою покладено раціональне поєднання та практичне використання властивостей регулярності. Така властивість притаманна основним елементам, з яких може бути побудована система керування технологічним обладнанням:

- об’єктам управління дискретної циклічної дії ($ТА_{ДЦ}$),
- технологічним мовам опису алгоритмів управління $ТА_{ДЦ}$;
- матричним мікроелектронним структурам [14. 15].

Розглянемо основні принципи функціонування і архітектуру керуючих автоматів з паралельним принципом керування і ПЛК, що побудовані на їх базі [3, 4].

1.1.1 Класичний ПЛІС-контролер паралельної дії

На рисунку 1.1 показано структуру сучасного ПЛІС-контролера паралельної дії. Показана структура покладена в основу промислових зразків ПЛК ПД і реалізована на кристалах ПЛІС компанії Altera [3].

Розглянемо елементи з яких будується ПЛК. Його основу складає чотири блоки пам’яті:

- БПС – блок пам'яті станів;
- БПК блок пам'яті команд;
- БПП – блок пам'яті переходів;
- БПЗК блок пам'яті заборонених комбінацій.

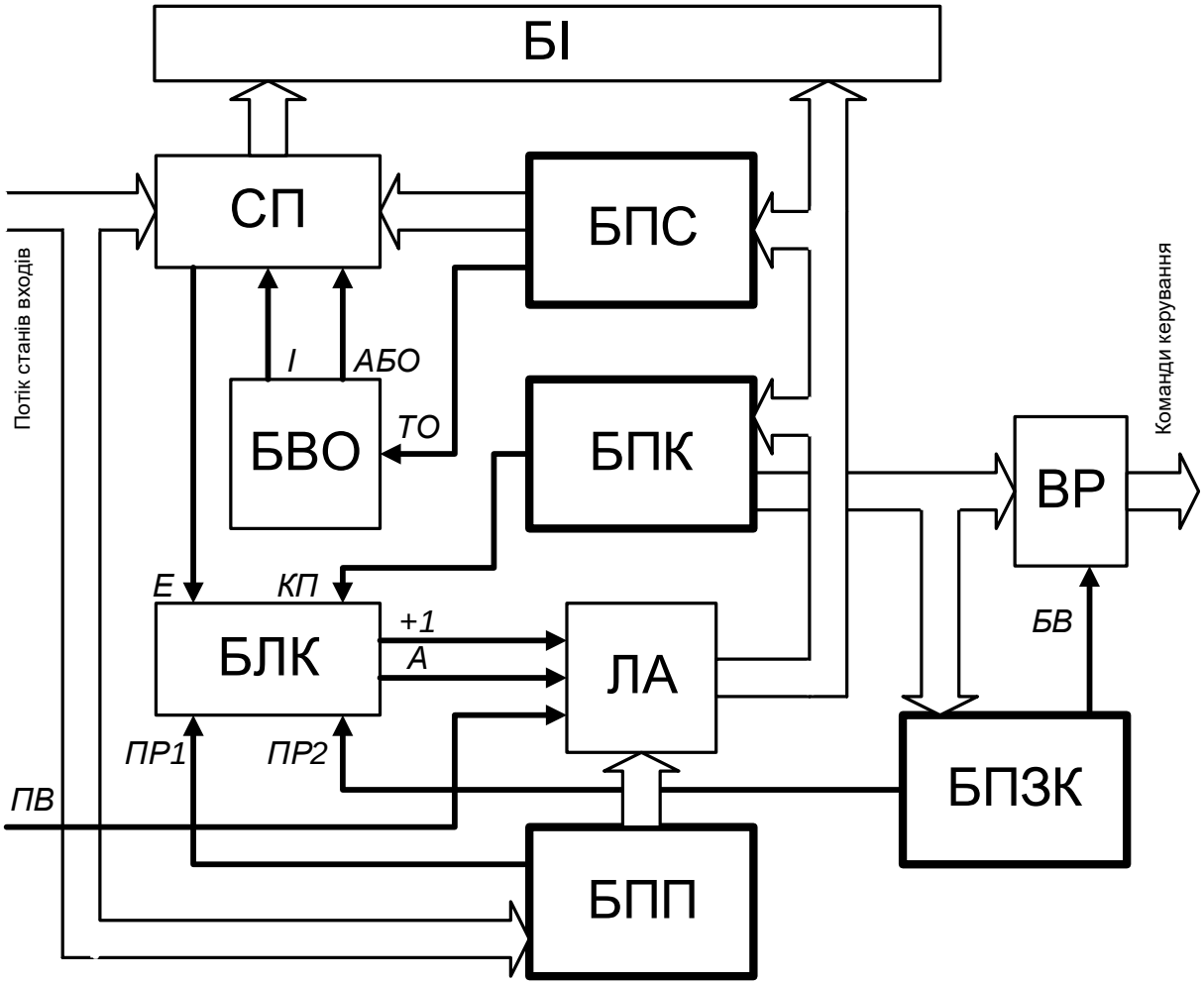

Рисунок 1.1 – Структура ПЛІС-контролера паралельної дії

Призначення блоку БПС – збереження очікуваних станів зовнішнього середовища. БПС містить у собі набір команд керування технологічним агрегатом. Блок БПП призначений для зберігання адрес переходів до підпрограм, що визначаються станом стохастичних входів. До БПЗК записано комбінації команд керування, видача яких на ТА є неприпустимою і має бути обов’язково блокована.

Також ПЛК ПД включає в себе блоки, що забезпечують логіку роботи, інтерфейсні функції та функції індикації:

- БІ – блок індикації;
- СП – схема порівняння;
- БВО – блок вибору операції;

- БЛК – блок логічного керування;
- ЛА – лічильник адреси;
- ВР вихідний регістр.

БЛК, як блок що визначає базову логіку роботи автомату, реалізує функції логічного керування за рахунок перевірки істинності рівнянь (1.1) і (1.2):

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

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

де КП – ознака кінця підпрограми;

ПР1 – ознака переривання від блоку пам’яті переходів;

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

А – початкова адреса підпрограми;

Е – сигнал еквівалентності;

+1 – сигнал переходу до наступного рядка.

Розглянемо реалізацію вказаних рівнянь на практиці. Реалізація рівняння (1.1) відбувається у разі появи на вході блоку пам’яті переходів однієї із запрограмованих комбінацій *(КП=А)*, що відповідає за вибір початкової адреси підпрограми. У такому випадку БПП встановлює лічильник адреси у відповідний даній комбінації стан. БЛК формує сигнал «*А*» (Адреса), за яким лічильник адреси переадресовує БПС і БПК на перший рядок обраної підпрограми. Для реалізації технічної можливості виконання такого переходу в останньому рядку кожної підпрограми і у нульовому рядку основної програми записується тільки ознака кінця підпрограми «*КП*». Ця ознака використовується як дозвіл переходу керуючого автомата до відпрацювання будь-якої із записаних у блоки пам'яті підпрограм.

Далі необхідно розглянути власне відпрацювання обраної підпрограми, що відбувається за рахунок реалізації рівняння (1.2). Формування блоком

БЛК сигналу «*+1*» відбувається у разі виконання цієї рівності. За наявності сигналу «*+1*» відбувається адресує лічильником адреси блоків пам’яті БПС і БПК на наступний (*i+1*) рядок поточної підпрограми. Умовою формування сигналу «*+1*» є наявність сигналу еквівалентності «*E*», який може бути сформований двома способами. Якщо на певному кроці керуючої програми необхідно порівнювати фактичний стан усіх датчиків циклу з їх очікуваними значеннями, тобто необхідно реалізувати логічну операцію «*І*», то до останнього стовпця і-го рядка, БПС, записується відповідна ознака і БВО формує сигнал «*І=1*». Цей сигнал вказує схемі порівняння, що необхідно реалізувати саме логічну операцію «*І*». У такому випадку сигнал еквівалентності *E* на її виході з'явиться лише у разі збігу усіх фактичних станів датчиків циклу з їх очікуваними значеннями, записаними до і-го рядку блоку пам’яті станів. Якщо перехід на наступний крок підпрограми можливий у випадку спрацювання лише одного датчика, то БВО формує сигнал «*АБО=1*», у такому випадку схема порівняння реалізує логічну операцію «*АБО*».

Ознаки переривань «*ПР1*», «*ПР2*», сигнали «*ПВ*» і «*БВ*», призначені для технічної реалізації безпекових функцій контролера і у контексті поточного наукового дослідження не розглядатимуться, але вони є необхідними з технічної точки зору. Більш детально описом роботи ПЛІС-контролера ПД можна ознайомитись у роботах [3, 4, 10].

Розглядаємий класичний варіант ПЛІС-контролера паралельної дії, що реалізує інформаційну технологію паралельного логічного керування, було практично реалізовано на Харківському державному виробничому об’єднанні «Приладобудівний завод ім. Т. Г. Шевченка». зовнішній вигляд контролера показано на рисунку 1.2 ПЛІС-контролер реалізовано на кристалі ПЛІС із завантажувальною Flash пам’яттю EPM1270T144 сімейства MAX II в упакуванні TQFP із 144 виводами.

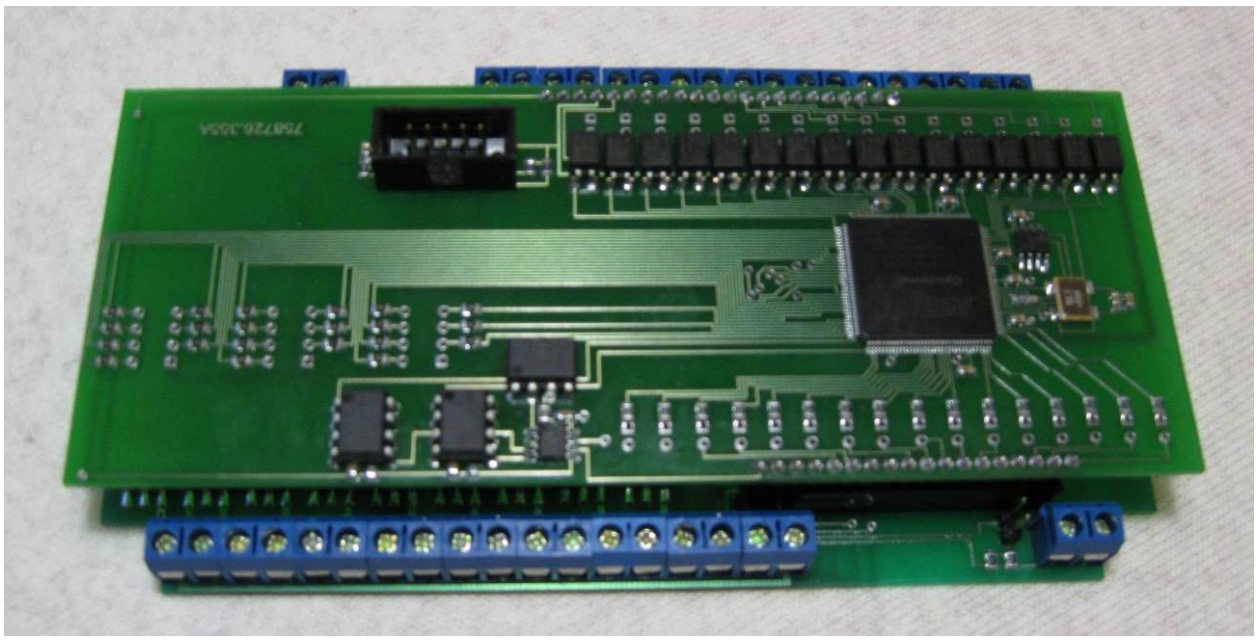

Рисунок 1.2 – Загальний вигляд монтажних плат промислового зразка ПЛІС-контролера паралельної дії

1.1.2 ПЛІС-контролер ПД із розширеними функціональними можливостями

Як було показано в [1] одним із перспективних напрямків розвитку керуючих пристроїв з паралельною архітектурою є вдосконалення їхньої внутрішньої організації. Наслідком цього можливе покращення безпекових можливостей керуючих систем, що побудовані на їх базі. Також це може розширювати можливості інструментів логічної обробки вхідних сигналів [3. 4], дозволяє вводити можливості реалізації функцій нечіткого логічного висновку [12, 13]. За результатами досліджень інформаційної технології паралельного логічного керування, а саме керуючих архітектур паралельної дії, було визначено пріоритетність уведення до таких структур внутрішніх програмованих користувачем таймерів і лічильників.

Розгляд структури ПЛІС-контролера паралельної дії, що був проведений вище, показав, що, для реалізації функцій програмованих таймерів в таких пристроях, необхідно внести зміни до обох рівнянь (1.1) і

(1.2). Тобто для повноцінної реалізації функції таймерів, як для переходу на наступний рядок поточної підпрограми, так і для переходу до іншої підпрограми мають бути модифіковані обидва рівняння, що описують роботу структури контролера. Таким чином для реалізації функцій програмованих таймерів необхідно внести зміни до деяких елементів структури ПЛІС-контролера і зв'язків між ними (рисунок 1.1), що приймають участь, як у формуванні сигналів «*+1*», так і сигналу «*КП*».

На рисунку 1.3 показано структуру ПЛІС-контролера паралельної дії із розширеним функціональними можливостями у якій реалізовано функції внутрішніх програмованих користувачем таймерів.

Розглянемо основні відмінності і особливості функціонування структури з елементами реалізації внутрішніх програмованих таймерів, у порівнянні з класичною структурою.

Основною відмінністю вдосконаленої структури є поява додаткового блоку пам'яті – блоку пам'яті внутрішніх змінних (БПВЗ). Цей блок пам'яті призначений для запам'ятовування чисельних значень внутрішніх таймерів. що програмуються користувачем. Також у структурі з'явився ще один додатковий функціональний блок – блок внутрішніх таймерів (БВТ). Саме БВТ дозволяє інтегрувати логіку роботи з внутрішніми таймерами до загальної логіки роботи ПЛІС-контролера ПД. Виведення блоку БВТ із загальної логіки роботи блоку логічного керування (БЛК) дозволяє спростити загальний опис роботи контролера і виокремити логіку, що пов'язана безпосередньо з таймерами.

Розглянемо призначення додаткових сигналів і блоків, що відрізняють вдосконалену структуру від базової.

Основними додатковими сигналами є сигнали ініціатори запуску відліку внутрішніх таймерів є «*ІН1*» та «*ІН2*». Значення цих сигналів записано до *i*-го рядку блоку пам'яті команд (БПК), або блоку пам'яті переходів (БПП) у вигляді елемента відповідного стовпчика. Тобто ці сигнали по своїй суті аналогічні сигналам від датчиків.

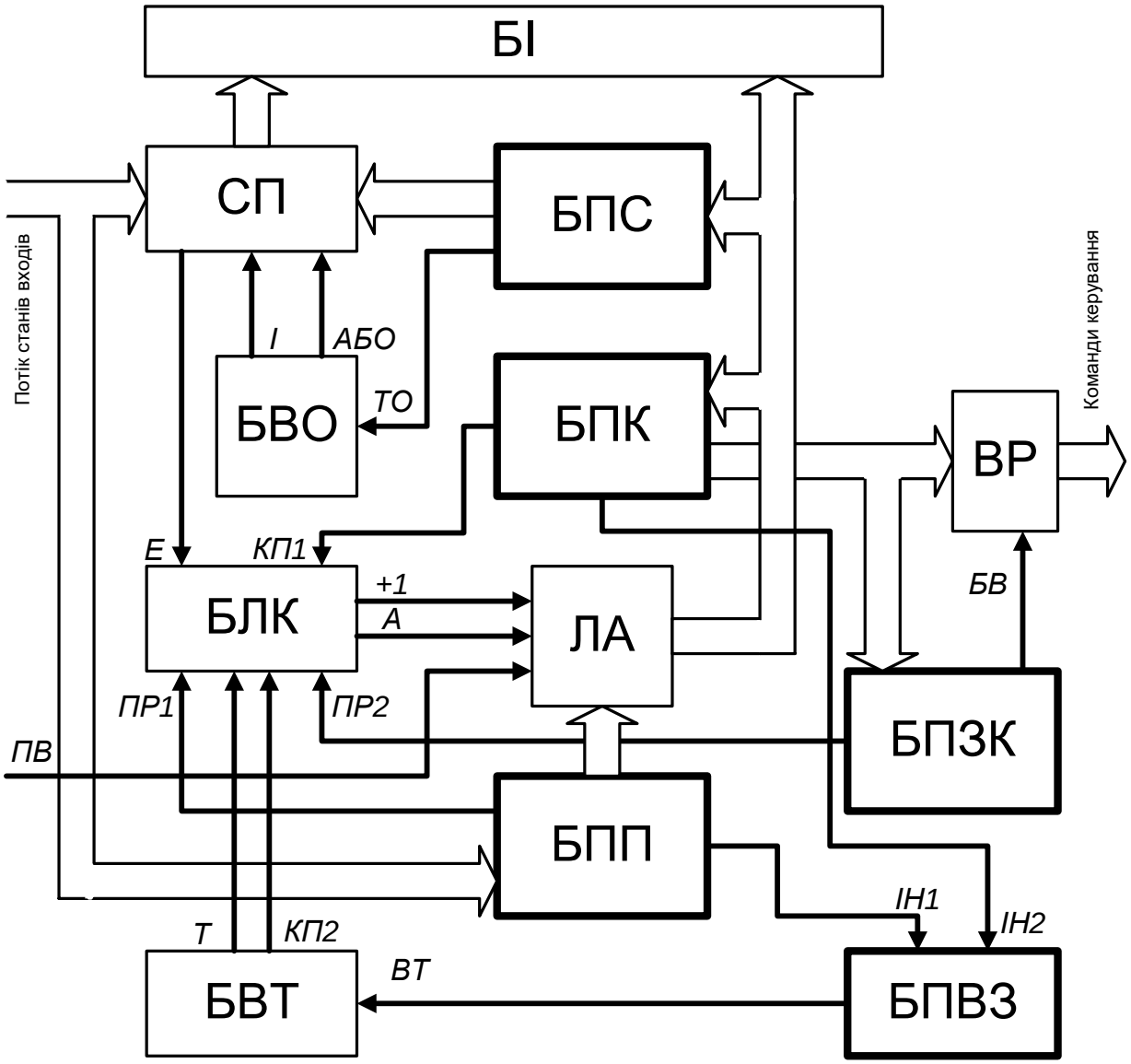

Рисунок 1.3 – Структура ПЛІС-контролера ПД з елементами реалізації внутрішніх програмованих таймерів

У момент часу закінчення формування інтервалу відліку таймера, блок пам’яті внутрішніх змінних формує сигнал «*ВТ*». Цей сигнал має наступне логічне навантаження: внутрішній таймер завершив свою роботу і запрограмований проміжок часу сформовано. За наявності сигналу «*ВТ*» блок внутрішніх таймерів ініціює формування, або сигналу «*Т*», або «*КП2*» у залежності від того, який саме блок (БПК, або БПП) був ініціатором формування внутрішнього відліку часу.

Сигнал «*T*» надає інформацію блоку БЛК про те, що необхідно сформувати сигнал «*+1*», тобто перехід до наступного *i+1*-го рядка поточної підпрограми. Тобто логіка роботи з внутрішнім таймером у частині, що відповідає за процес відпрацювання поточної підпрограми, фактично відповідає логіці обробки сигналів детермінованих входів. Але є і певні відмінності. При роботі із сигналами детермінованих входів схема порівняння «очікує» на появу єдиної «дозволеної» комбінації вхідних сигналів, а при роботі з таймером «єдина дозволена» комбінація має сформуватись «автоматично» через певний проміжок часу, що визначається при програмуванні таймерів.

Формально сигнал «*+1*» формуватиметься у відповідності до наступного рівняння:

$$+1 = (E \vee T) \wedge \overline{КП1} \wedge \overline{КП2} \wedge \overline{ПР1} \wedge \overline{ПР2} \qquad (1.3)$$

Розглянемо відмінності логіки роботи контролера при переході до іншої підпрограми. У такому випадку ініціатором відліку стає блок БПП і, відповідно, блок БВТ ініціює формування сигналу «*КП2*». Логіка роботи з сигналом «*КП2*» повністю відповідає логіці роботи з сигналом «*КП*» (кінець підпрограми) класичної структури. У розглядаємій структурі сигнал «*КП*» отримав додатковий індекс – «*КП1*». Таким чином перехід до іншої 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.

Таким чином формування сигналу «*A*» у структурі ПЛІС-контролера з таймерами відбувається у відповідності до рівняння:

$$A = КП1 \vee КП2 \vee ПР1 \vee ПР2 \quad (1.4)$$

Необхідно також зазначити, що власне 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 контролері. Реалізація процедури відліку виконується незалежними апаратними засобами кристалу ПЛІС. Тобто можна зробити висновок про те, що виконання процедури відліку проміжку часу програмованим таймером ніяким чином не залежить від внутрішньої логіки роботи ПЛІС-контролера.

1.2 Постановка завдання дослідження

Проведений аналіз показує, що керуючі пристрої з паралельною архітектурою, мають значні переваги у порівнянні з класичними системами послідовної дії. Серед таких необхідно вказати дві найважливіші. перша – це практично повна відсутність залежності кількості контрольованих входів і керованих виходів від швидкодії контролера. Друга перевага – це можливість реалізації TVP-технології для автоматизованого створення керуючих програм неспеціалістом в області програмування і за допомогою спрощених мов. Як було показано у [19], використання такої технології дозволяє значно зменшити кількість помилок у програмному коді та пришвидшує процес його формування. Ще одним, на думку автора найважливішим аспектом, є 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 дуже актуальною задачею.

Враховуючи вищевказане, основною метою роботи є розробка HDL-

моделі логічного керуючого автомату паралельної дії з розширеним функціоналом у вигляді програмованих таймерів, для формування можливості його практичної реалізації на базі ПЛІС.

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

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

- розглянути існуючі HDL-моделі ЛКА ПД на базі ПЛІС, визначити шляхи інтеграції до існуючих моделей елементів, що реалізують додатковий функціонал;

- провести аналіз існуючих систем автоматизованого проектування для роботи з кристалами ПЛІС, визначити пріоритетні САПР для реалізації HDL-моделі ЛКА ПД;

- розробити HDL-модель ЛКА ПД, що містіть у собі елементи для реалізації програмованих таймерів і задовольняє вимогам реалізації фізичного прототипу ПЛІС-контролера нового покоління.

2 HDL-МОДЕЛЬ ПЛІС-КОНТРОЛЕРА ПАРАЛЕЛЬНОЇ ДІЇ

2.1 ПЛІС, як перспективний засіб побудови керуючих пристроїв з паралельною архітектурою

Програмовані логічні інтегральні схеми (ПЛІС) є інтегральними схемами високого рівня інтеграції для розробки на їх основі спеціалізованих пристроїв. ПЛІС поєднують переваги масового заводського виробництва ІМС з можливістю гнучкого підбору структури схеми для малосерійних спеціалізованих пристроїв. на базі ПЛІС можуть бути виготовлені логічні блоки та системи, перетворювачі кодів, периферійні контролери, мікропрограмні пристрої керування, кінцеві автомати, помножувачі, невеликі процесори, в тому числі для швидкого перетворення Фур'є. Проектування пристрою на ПЛІС полягає у складанні схеми з’єднань логічних елементів, що входять до її складу, та подальшому програмуванні матриці спеціальним програматором, що може бути виконане самим користувачем. Основна перевага ПЛІС перед іншими спеціалізованими схемами – мінімальний час виготовлення необхідних замовних варіантів схем. Зникає необхідність звертатися до виробників ІМС для нанесення металевої маски та установки кристала у корпус. Час отримання потрібної схеми із стандартної ПЛІС вимірюється секундами та хвилинами. Широке поширення ПЛІС стало можливим завдяки наявності безлічі автоматизованих систем проектування, що було розроблено різними виробниками для вирішення своїх завдань.

У більшості випадків мікросхема ПЛІС складається з набору реконфігурованих логічних блоків, що реалізують необхідну логічну функцію; програмованих перемикачів, що здійснюють потрібне з’єднання логічних осередків; програмованих блоків введення/виведення, що забезпечують зв’язок зовнішнього виведення мікросхеми з внутрішньою структурою ПЛІС. У сучасних ПЛІС часто бувають вбудовані додатково

блоки пам'яті, DSP (Digital Signal Processor – цифровий сигнальний процесор) або помножувачі, PLL (Phase Locked Loop – фазовий автопідстроювач частоти) та інші компоненти.

Якщо розробляється певний проект, його розробник може, як правило, не враховувати особливості елементів, з яких складається ПЛІС. При цьому він описує бажані функції проектованої мікросхеми або записує текст у системі програмного забезпечення (наприклад, мовами опис апаратури Verilog або VHDL). Необхідну схему конструює компілятор на основі відомої внутрішньої структури ПЛІС, який сам розміщує елементи схеми за наявними конфігурованими логічними блоками і з’єднує ці блоки за допомогою наявних програмованих ліній зв'язку.

Мови опису апаратури завойовують все більшу популярність серед розробників. У великій мірі цьому сприяла свого часу популярність САПР MAX+PLUS II BASELANE фірми Altera, який підтримує створення описів трьома мовами: AHDL, VHDL і Verilog. З моменту появи MAX+PLUS II із цих трьох мов саме мова AHDL захопила і, за деякими даними, до цього дня утримує перше місце за популярністю у інженерів-розробників на пострадянському просторі. Причин тому можна вказати дві. Перша – це надзвичайна простота і чудова документованість у довідниковій системі, що особливо приваблює інженерів старшого покоління. Друга важлива причина це непоганий асортимент літератури з мови AHDL. З літературою з мови VHDL, нажаль, ситуація не така хороша.

AHDL (AlteraHDL) є внутрішньою мовою пакета MAX+PLUS II. Це означає, що в якому б вигляді не виконувався проект (графічний, AHDL, VHDL, Verilog або у вигляді тимчасових діаграм у Waveform Editor), компілятор MAX+PLUS II спочатку перекладе проект на мову AHDL. Потім компілятор перетворить AHDL-файл на файл прошивки з розширенням «.pof».

2.2 HDL-модель класичного ПЛІС-контролера паралельної дії

Для практичної реалізації ПЛІС-контролера паралельної дії, за структурою що показана на рисунку 1.1 побудовано HDL-модель ЛКА ПД (рисунок 2.1). Така модель дозволяє виконати перехід від абстрактних моделей і структур до практичного втілення таких моделей спеціалізованими інструментальними засобами розробки і моделювання кристалів ПЛІС. HDL-модель ЛКА ПД орієнтована на реалізацію мовою AHDL і виконана і промодельована за допомогою САПР MAX+PLUS II BASELANE.

Для розуміння принципів побудови і функціонування HDL-модель класичного ЛКА ПД необхідно розглянути функціональні блоки і компоненти блоків. Також розглянемо їх призначення, виконувані функції та результати моделювання їхньої роботи. Такий розгляд дозволить перейти до побудови HDL-моделі ПЛІС-контролера паралельної дії із розширеними функціональними можливостями, структуру якого наведено на рисунку 1.3 першого розділу роботи.

Функціональний блок *bps:_bps* – блок пам’яті станів. Цей функціональний блок призначений для зберігання комбінації станів детермінованих входів, що є диною «дозволеною» для включення відповідної кінцевої комбінації механізмів. Відповідно до розглянутої раніше структури, блок ***bps:_bps*** реалізує його блок БПС. ***bps:_bps*** містить *q*-розрядний вхід адреси *adr[q..1]* та *k*-розрядний вихід двобітних даних *vec_Ai[1..k][1..2]*. Двійковий вихід *ot* призначений для вбору типу операції. На цьому виході формується сигнал логічного «0» або «1» у залежності від типу логічної операції «І» або «АБО», що має виконуватись СП.

Функціональний блок *bpk:_bpk* – блок пам’яті команд. Блок призначений для зберігання команд керування $ТА_{ДЦ}$. Блок ***bpk:_bpk*** має *q*-розрядний вхід адреси *adr[q..1]* та *m*-розрядний вихід даних *vec_c[1..m]*. Для реалізації сигналу «кінець підпрограми» блоком, призначений двійковий вихід *endsp*. Цей вихід приймає значення логічної «1», при наявності у

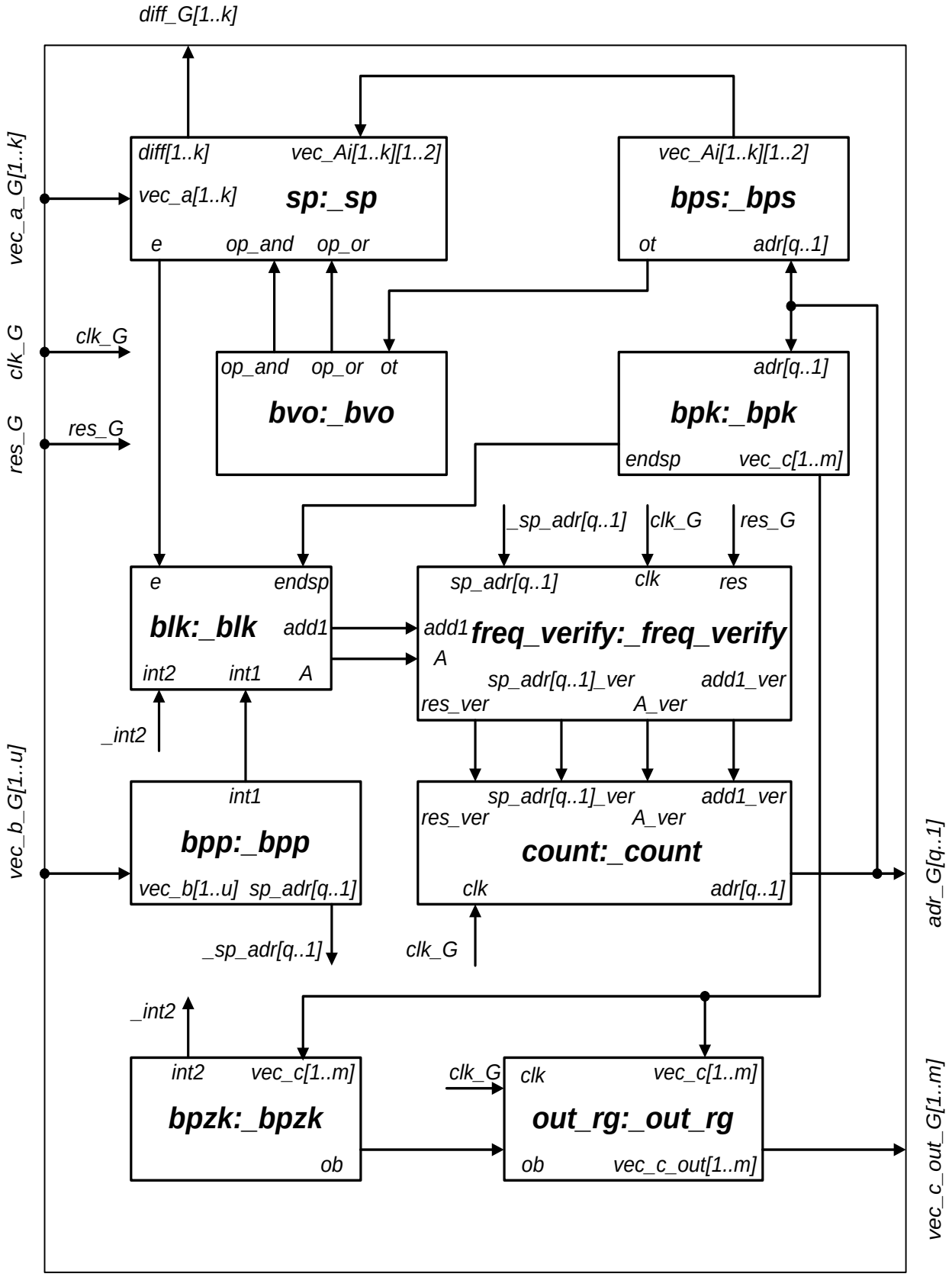

Рисунок 2.1 – HDL-модель класичного ЛКА ПД

векторі даних ознаки кінця підпрограми «КП», що дозволяє виконати перехід до виконання іншої підпрограми у програмах з розгалуженнями. У класичній структурі вихід «КП» лише один. Блок ***bpk:_bpk*** призначений для реалізації блоку БПК структури ЛКА ПД.

Функціональний блок *bpp:_bpp* – блок пам'яті переходів. Цей блок забезпечує зберігання і формування початкової адреси наступної підпрограми. Блок ***bpp:_bpp*** призначений для реалізації у кристалі ПЛІС блку БПП структури ЛКА ПД, він містить *u*-розрядний вхід даних *vec_b[1..u]* та *q*-розрядний вихід адреси початку підпрограми *sp_adr[q..1]*. У випадку наявності у вхідному векторі *vec_b[1..u]* ознаки аварійного переривання (аварійне переривання 1), на виході блоку ***bpp:_bpp*** *int1* формується сигнал логічної одиниця, що призводить до миттевого переходу до відпрацювання підпрограми аварійного перермивання, без очікування закінчення відпрацювання поточної підпрограми.

Функціональний блок *bpzk:_bpzk* – блок пам'яті заборонених комбінацій. По аналогії з попередніми блоками, призначений для реалізації засобами кристала ПЛІС блока БПЗК структури ЛКА ПД. Блок призначений для запам'ятовування заборонених комбінацій команд керування, видача яких на ТА є неприпустимою. Блок ***bpzk:_bpzk*** має *m*-розрядний вхід даних та два виходи – *int2* (вихід аварійного переривання 2) і *ob* , що призначений для блокування видачі на вихід забороненої комбінації вихідних сигналів на вході *vec_c[1..m]*.

Функціональний блок *sp:_sp* – призначений для виконання операції логічного порівняння сигналів на *k*-розрядних входах даних *vec_a[1..k]* і *vec_Ai[1..k][1..2]*. Для забезпечення виконання порівняння за логікою «І» або «АБО» призначені входи блоку *op_and* і *op_or* . У залежності від обраної операції порівняння сигнал логічної «1» на виході еквівалентності *e* буде сформовано або при порозрядному співпадінні сигналів векторів *vec_a[1..k]* і *vec_Ai[1..k][1..2]*, або при співпадінні сигналів хоча б у одному із розрядів вхідних векторів. Сигнали на *k*-розрядному виході *diff[1..k]* блоку ***sp:_sp***

формуються як результат операції порівняння і можуть бути використані для виводу на блок індикації (БІ). На рисунку 2.2 показано результати моделювання роботи блоку ***sp:_sp*** у процесі виконання порівняння.

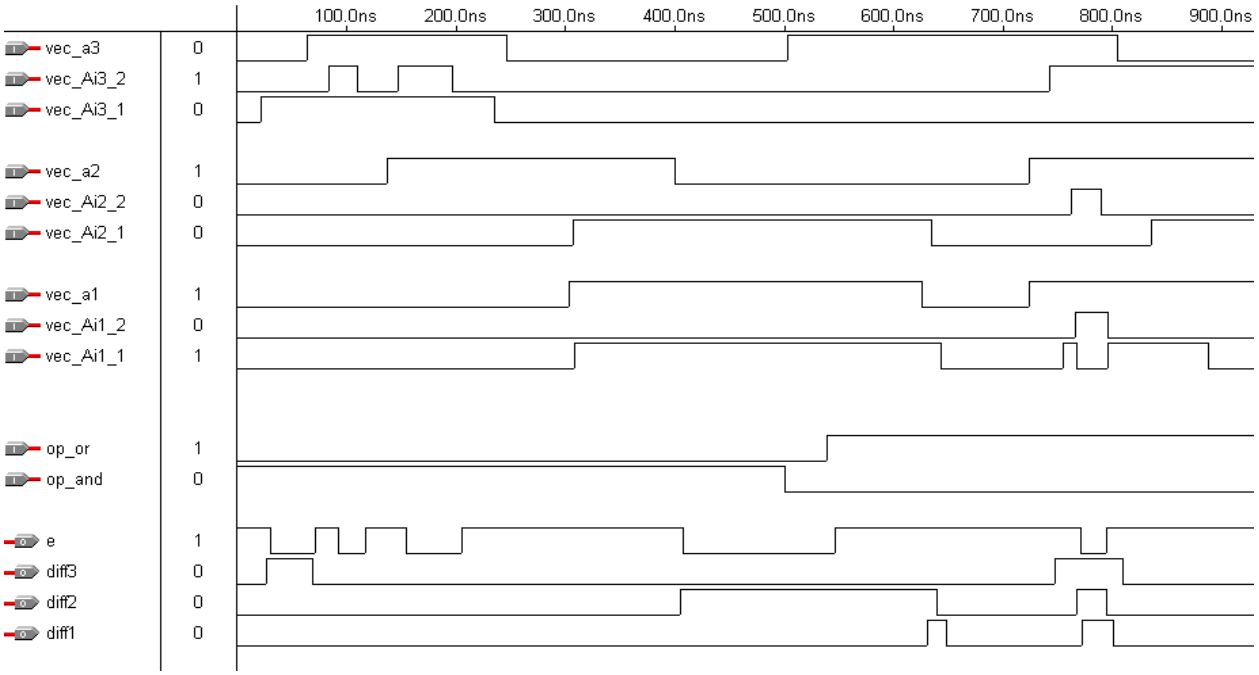

Рисунок 2.2 – Результат моделювання роботи блоку ***sp:_sp***

Функціональний блок *bvo:_bvo*. Як було вказано раніше, при описі роботи функціонального блоку ***sp:_sp***, порівняння сигналів детермінованих входів із запрограмованими до БПС значеннями може виконуватись двома способами. Саме для забезпечення цієї можливості призначений функціональний блок ***bvo:_bvo***, що забезпечує формування команд вибору типу операції для блоку логічного порівняння СП. Він має двійковий вхід *ot* (вхід вибору типу операції) і двійкові виходи *op_and* (приймає значення логічної «1» при виборі операції «І») і *op_or* (приймає значення логічної «1» при виборі операції «АБО»).

Функціональний блок *count:_count* – блок реалізації лічильника адреси (ЛА). Блок забезпечує адресацію блоків БПС і БПК. Блок має *q*-розрядні вхід *sp_adr_ver[q..1]* (вхід адреси переходу) і вихід *adr[q..1]*. Для

забезпечення коректності функціонування ЛКА ПД у цілому призначені додаткові входи синхронізації *clk*, керування *res_ver* (вхід команда скиду), *A_ver* (вхід команди паралельного завантаження) і *add1_ver* (вхід команди додавання одиниці). Результат моделювання роботи блоку можна розглянути на рисунку 2.3.

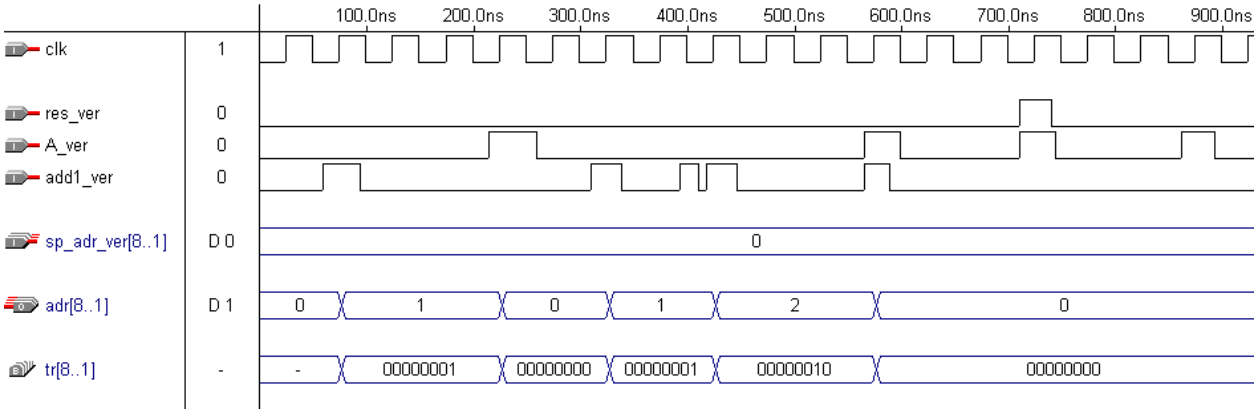

Рисунок 2.3 – Результат моделювання роботи функціонального блоку ***count:_count***

Функціональний блок *blk:_blk*. Блок логічного керування – це блок, що призначений для формування загальної логіки роботи ЛКА ПД. Блок забезпечує керування лічильником адреси. тобто, фактично, формування команди переходи до наступного кроку підпрограми, або переходу до іншої підпрограми. Це залежить від сигналів, що сформовані суміжними функціональними блоками ЛКА ПД. Блок ***blk:_blk*** має наступні входи і виходи:

- логічний вхід *e* (еквівалентність);

- логічний вхід *endsp* (кінець підпрограми);

- логічні входи *int1*, *int2* (виходи формування сигналів аварійного переривання 1 і 2);

- вихід керування лічильником адреси – *A* (паралельне завантаження адреси);

- вихід керування лічильником адреси *add1* (додавання одиниці). Результати моделювання роботи функціонального блоку ***blk:_blk*** показано на рисукнку 2.4.

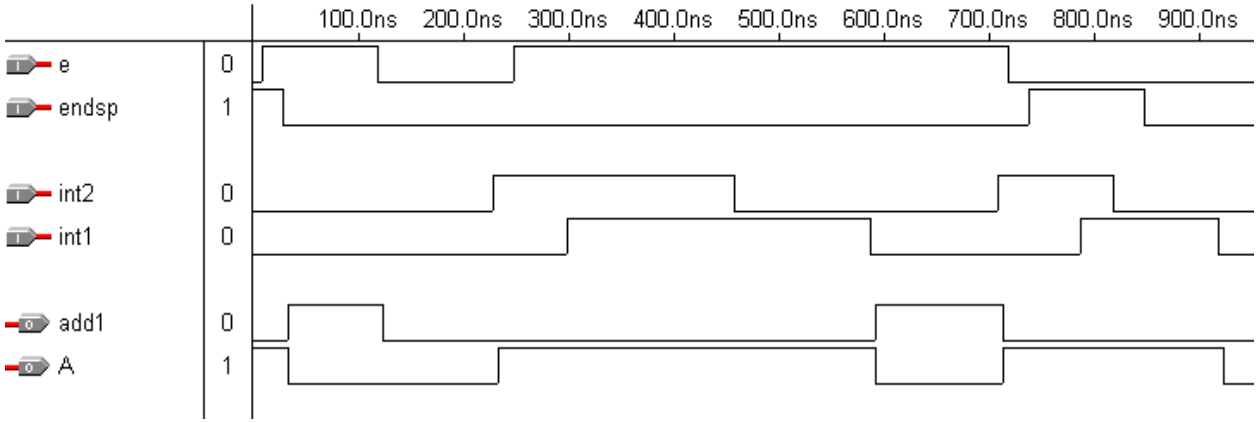

Рисунок 2.4 – Результати моделювання роботи функціонального блоку ***blk:_blk***

Функціональний блок *out_rg:_out_rg* – блок вихідного регістру. Блок ***out_rg:_out_rg*** призначений для тимчасового зберігання і видачі команд керування на технологічний агрегат. У випадку наявності забороненої комбінації команд керування, що сформовані блоком пам'яті команд (наприклад у випадку критичного збою або пошкодження структури ЛКА ПД), відбувається блокування видачі такої комбінації на ТА. Блок має *m*-розрядний вхід *vec_c[1..m]* і вихід *vec_c_out[1..m]* даних, також присутні входи синхронізації – *clk*, і вхід блокування виходу – *ob*.

Функціональний блок багатократного контролю *freq_verify:_freq_verify*. У випадку реалізації конкретного промислового зразка ПЛІС-контролера ПД, блок забезпечує 8-ми кратну (а у загальному випадку *n*-кратну) перевірку достовірності сигналів керування логікою роботи ЛКА ПД (сигналів БЛК, скиду ЛКА і сигналів переходу до підпрограм). Структурна схема блоку ***freq_verify:_freq_verify*** показана на рисунку 2.5. Результати моделювання його роботи показано на рисунку 2.6.

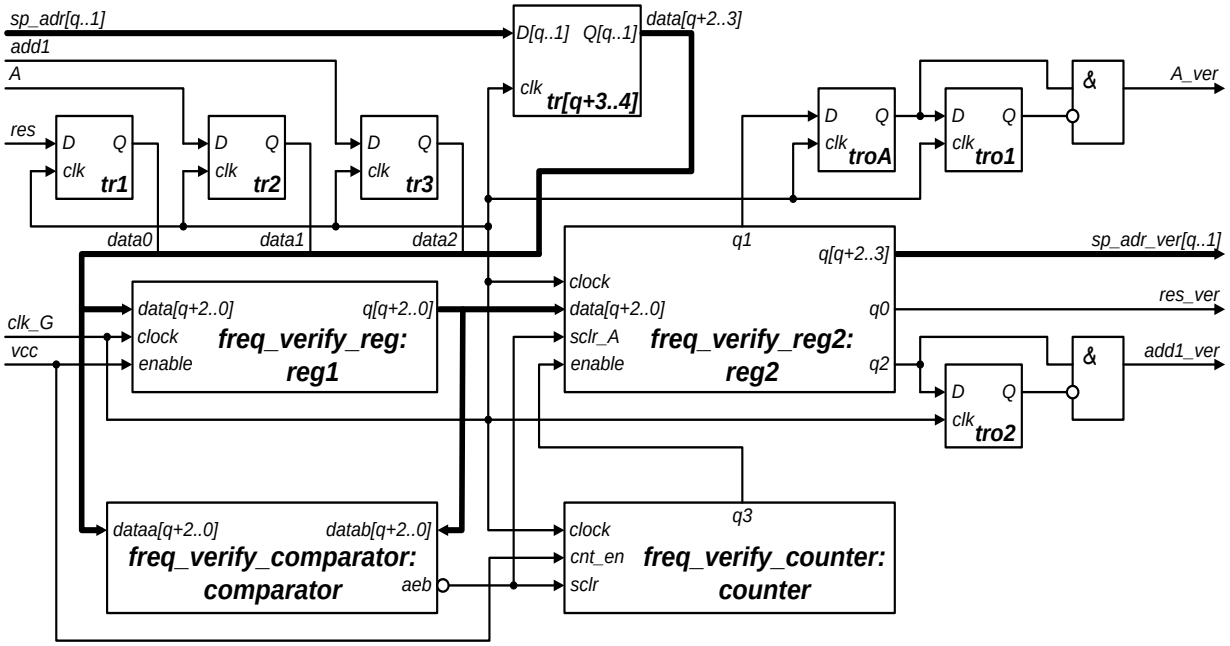

Рисунок 2.5 – Структурна схема функціонального блоку багатократного контролю ***freq_verify:_freq_verify***

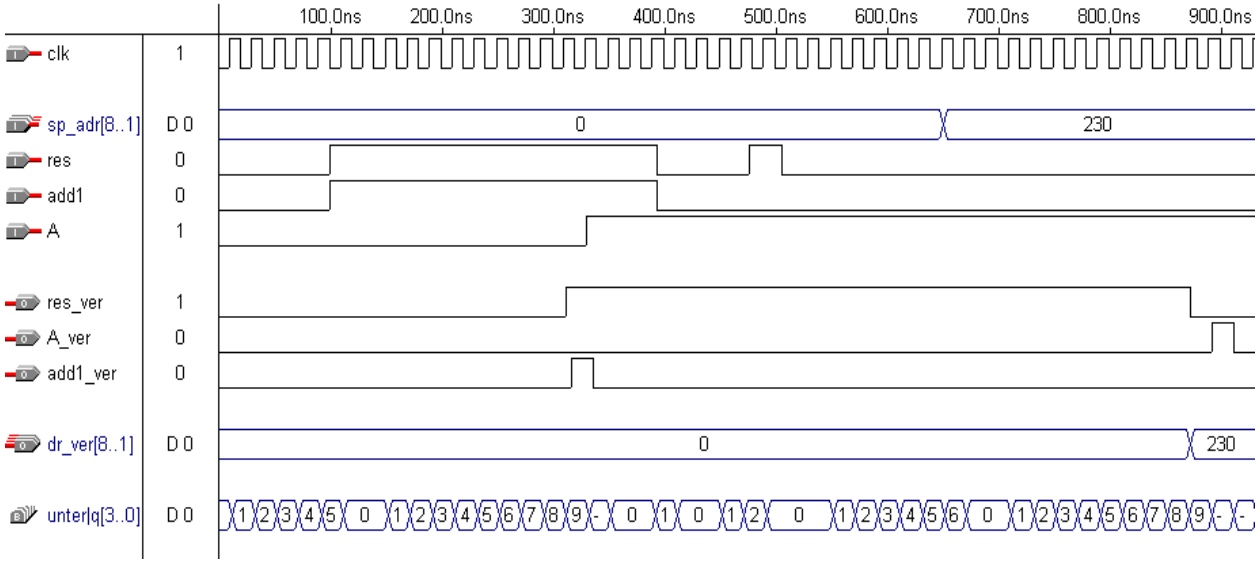

Рисунок 2.6 – Результат моделювання роботи функціонального блоку *freq_verify:_freq_verify*

Для розуміння призначення і логіки функціонування блоку ***freq_verify:_freq_verify*** розглянемо його складові компоненти та входи-виходи.

Компонент *freq_verify_reg:reg1* – (*q+3*)-и розрядний паралельний регістр. Компонент має наступні входи-виходи:

- вхід *clock* (сигнал синхронізації);
- вхід *enable* (вхід дозволу запису інформації);
- інформаційні входи *data[q+2..0]*;
- інформаційні виходи *q[q+2..0]*.

призначення трьох молодших розрядыв регістру полягає у тому, що вони використовуються для зберігання сигналів *res, A, add1* (сигналів скиду, адреси переходу та додавання одиниці). У той час, як усі інші *q*-розрядів, призначені для зберігання адреси переходу – *sp_adr[q..1]*.

Компонент *freq_verify_reg2:reg2* – (*q+3*)-и розрядний паралельний регістр. Цей регістр призначений для фіксації і тимчасового зберігання достовірної комбінації сигналів *res, A, add1* і *sp_adr[q..1]*. Конструктивно цей компонент побудовано аналогічно компоненту ***freq_verify_reg:reg1***, на відміну від якого містить додатковий вхід *sclr_A* – вхід асинхронного скиду 1-го розряду регістра (розряду команди *A_ver*). На старших *q*-розрядах регістра формуються достовірні сигнали адреси *sp_adr_ver[q..1]*. а Молодший розряд регістру *q0* призначений для зняття достовірної команди *res_ver* – команди скиду контролера.

Компонент *freq_verify_comparator:comparator* – компонент, що виконує додаткові функції схеми порівняння. Цей компонент призначений для фіксації зміни станів вхідних сигналів. Він має інформаційні входи *dataa[q+2..0], datab[q+2..0]* і один інверсний вихід *aeb*.На цьому виході з’являється сигнал логічної «1» якщо сигнали *dataa[q+2..0]* і *datab[q+2..0]* є нееквівалентними.

Компонент *freq_verify_counter:counter* – 4-х розрядний лічильник. Має наступні входи-виходи:

- вхід *clock* (сигнал синхронізації);
- вхід *cnt_en* (вхід дозволу роботи);
- вхід *sclr* (вхід синхронного скиду лічильника);
- вихід *q3*.

Сигнал логічної одиниці на виході *q3* формується лише у випадку, якщо вхідні сигнали *data[q+2..0]* були незмінними протягом певного (заздалегідь визначеного) інтервалу часу.

Компонент *tr (tr1-tr[q+3])* – синхронні D-тригери. Ці тригери призначені для синхронізації видачі сигналів ЛКА ПД *res, A, add1* та *sp_adr [q..1]* із глобальним сигналом *clk_G*. Поява цього компонента визначається архітектурою побудови кристала ПЛІС і принципами його роботи. Тобто цей компонент не впливає на загальну логіку роботи ЛКА ПД. Компонент ***tr (tr1-tr[q+3])*** має входи даних *D*, синхронізації *clk* та вихід *Q*.

Компонент *tro (tro1, tro2)* – синхронні D-тригери. Ці тригерию.аналогічно до тригерів компоненту *tr*, сумісно з логічними елементами «І», призначені для формування достовірної команди *ver_A* і *ver_add1* довжиною в один період глобального сигналу синхронізації *clk_G*. Цей компонент також не впливає на загальну логіку роботи ЛКА ПД.

Компонент *troA* – синхронний D-тригер. По аналогії до компоненту *tr* призначений для затримки видачі команди ЛКА ПД *A_ver* на один період глобального сигналу синхронізації *clk_G*. Цей компонент так само не впливає на загальну логіку роботи ЛКА ПД і визначається лише архітектурою побудови кристала ПЛІС.

3 ІНСТРУМЕНТАЛЬНЫ ЗАСОБИ РОЗРОБКИ HDL-МОДЕЛЕЙ КЕРУЮЧИХ ПРИСТРОЇВ З ПАРАЛЕЛЬНОЮ АРХІТЕКТУРОЮ

3.1 САПР MAX+PLUS II BASELANE

Для проектування цифрових пристроїв на ПЛІС можуть бути використані різні системи автоматизованого проектування (САПР), які забезпечують можливість опису цифрових пристроїв, виконання компіляції, верифікації та безпосередньо програмування ПЛІС. Однією з найпоширеніших, у свій час, САПР була система MAX+PLUS II. Система MAX+PLUS II розроблена фірмою Altera та містить повний спектр можливостей логічного проектування: різноманітні засоби опису проектів з ієрархічною структурою, потужний логічний синтез, компіляцію із заданими часовими параметрами, автоматичний поділ проекту на частини, функціональне та тимчасове тестування, тестування кількох пов'язаних пристроїв, аналіз тимчасових параметрів системи, автоматичну локалізацію помилок, а також програмування та верифікацію пристроїв.

САПР MAX+PLUS II є інтегрованим середовищем для розробки цифрових пристроїв на базі програмованих логічних інтегральних схем (ПЛІС) фірми «Altera» і забезпечує виконання всіх етапів, необхідних для випуску готових виробів:

- створення проектів пристроїв;

- синтез структур та трасування внутрішніх зв'язків ПЛІС;

- підготовку даних для програмування чи конфігурування ПЛІС (компіляцію);

- верифікацію проектів (функціональне моделювання та часовий аналіз);

- програмування чи конфігурування ПЛИС.

До складу пакету MAX+PLUS II входять наступні пов'язані між собою

додатки, що реалізують усі вказані вище етапи розробки цифрових пристроїв на ПЛІС фірми «Альтера». Вкажемо їх, та розглянемо основне призначення і можливості.

Graphic Editor – графічний редактор, призначений для введення проекту у вигляді схеми з'єднань символів елементів, що витягуються зі стандартних бібліотек пакету або бібліотеки користувача.

Waveform Editor – редактор тимчасових діаграм (сигнальний редактор), що виконує подвійну функцію. На етапі введення забезпечується введення логіки проекту у вигляді діаграм (епюр) станів входів та виходів, а на етапі моделювання забезпечується введення діаграм тестових (еталонних) вхідних станів модельованого пристрою та завдання переліку тестованих виходів.

Text Editor – текстовий редактор, призначений для створення та редагування текстових файлів, що містять опис логіки проекту мовою опису пристроїв AHDL (Altera Hardware Description Language) або близькими до нього мовами типу VHDL, «Verilog».

Symbol Editor – символьний редактор, що дозволяє редагувати існуючі символи та створення нових. Будь-який відкомпільований проект може бути згорнутий у символ, розміщений у бібліотеці символів і використаний у якості елементу у будь-якому іншому проекті.

Floorplan Editor – редактор зв'язків (порівневий планувальник), який на плані розташування основних логічних елементів дозволяє вручну розподіляти висновки ПЛІС (закріплювати виводи за конкретними вхідними та вихідними сигналами) та перерозподіляти деякі внутрішні ресурси ПЛІС.

Compiler Netlist Extractor – додаток, що забезпечує вилучення списку з'єднань з вихідного файлу представлення проекту, створеного під час введення проекту.

Database Builder – додаток, призначений для побудови бази даних проекту.

Logic Synthesizer – додаток, що забезпечує перевірку коректності проекту з формальним правилам та синтез оптимальної структури проекту.

Partitioner – додаток, що забезпечує розбиття проекту на частини у тих випадках, коли ресурсів одного кристала (мікросхеми) замало для реалізації проекту.

Fitter – трасувальник внутрішніх зв'язків, що забезпечує реалізацію синтезованихної структури.

Timing SNF Extractor – додаток, що забезпечує вилучення параметрів проекту, що необхідні для функціонального моделювання та тимчасового аналізу.

Assembler – додаток, що створює з скомпілюваного проекту файли, які необхідні для програмування або іншими словами конфігурування кристалу ПЛІС за допомогою програматора.

Simulator – додаток, який спільно з редактором тимчасових діаграм призначено для функціонального моделювання проекту з метою перевірки правильності логіки його функціонування.

Timing Analyzer – додаток, що забезпечує розрахунок часових затримок від кожного входу до кожного логічно пов'язаного з ним виходу.

Для програмування або конфігурування кристалу ПЛІС використовується додаток MAX + PLUS II Programmer. Програмування та перепрограмування мікросхем, що мають вбудовану систему програмування (ISP), може здійснюватися безпосередньо у складі кінцевого виробу через спеціальний кабель, що підключається або до LPT-порту (Byte Blaster), або до COM-порту (Bit Blaster) комп’ютера та технологічного 10-контактного з’єднувача інтерфейсу JTAG, встановлюваного на платі виробу. Якщо на платі виробу встановлюється кілька ПЛІС із вбудованими системами програмування, то всі вони можуть програмуватися через один технологічний роз’єм. Для цієї цілі програма «Programmer» має режим «Multi-Device». Схеми підключення ПЛІС до інтерфейсу JTAG наведено у документації на «Bit Blaster» і «Byte Blaster». Для програмування інших мікросхем необхідно додатково використовувати зовнішній програматор, який також може підключатися доUSB, COM або LPT-порту.

До складу САПР MAX + PLUS II, крім того, входять ще три сервісні програми.

Design Doctor – додаток, призначений для перевірки коректності проекту з використанням емпіричних правил.

Message Processor – процесор повідомлень, що забезпечує обробку, висновок на відображення та локалізацію (вказівка місця у проекті, до якого воно відноситься) повідомлень трьох типів: повідомлень про помилки («Error»), попереджень («Warning») та інформаційних повідомлень («Info»). Причину виведення того чи іншого повідомлення можна з'ясувати через опцію Help on Message процесора повідомлень. За наявності повідомлень про помилки компіляція проекту неможлива до їх повного усунення. За наявності попереджень компіляція успішно завершується, проте наявність попередження свідчить про виявлення проблеми, що може призвести до неправильної роботи пристрою. Тому всі попередження повинні бути ретельно проаналізовані з використанням «Help on Message», до з'ясування причин їх появи та подальшого усунення цих причин (або ігнорування попередження, що іноді буває можливо). Інформаційні повідомлення потрібно лише брати до уваги.

Hierarhy Display – додаток, що забезпечує огляд ієрархічної структури проекту, що може складатися з безлічі складених у різних редакторах та згорнутих у символи проектів нижчих рівнів, причому кількість рівнів не обмежується. Основний проект (проект найвищого рівня) має бути створений у графічному редакторі (якщо проект має лише один рівень ієрархії, то він може бути створений у будь-якому редакторі).

Процедура виконання проекту в САПР MAX + PLUS II складається з послідовного виконання наступних етапів:

- видача технічного завдання;

- введення опису модулів у графічному, текстовому або редакторі часових діаграм;

- робота з компілятором:

1) перевірка синтаксису написання;

2)логічний синтез;

3) розведення кристалу ПЛІС;

4) побудова бази даних для часового моделювання;

5) створення файлу для програмування кристалу ПЛІС;

- аналіз швидкодії ПЛІС у часовому аналізаторі;

- часове моделювання у системі моделювання;

- програмування кристалу і перевірка його роботи у системі програмування;

- виготовлення серійного зразка.

Головне вікно САПР MAX + PLUS II показано на рисунку 3.1.

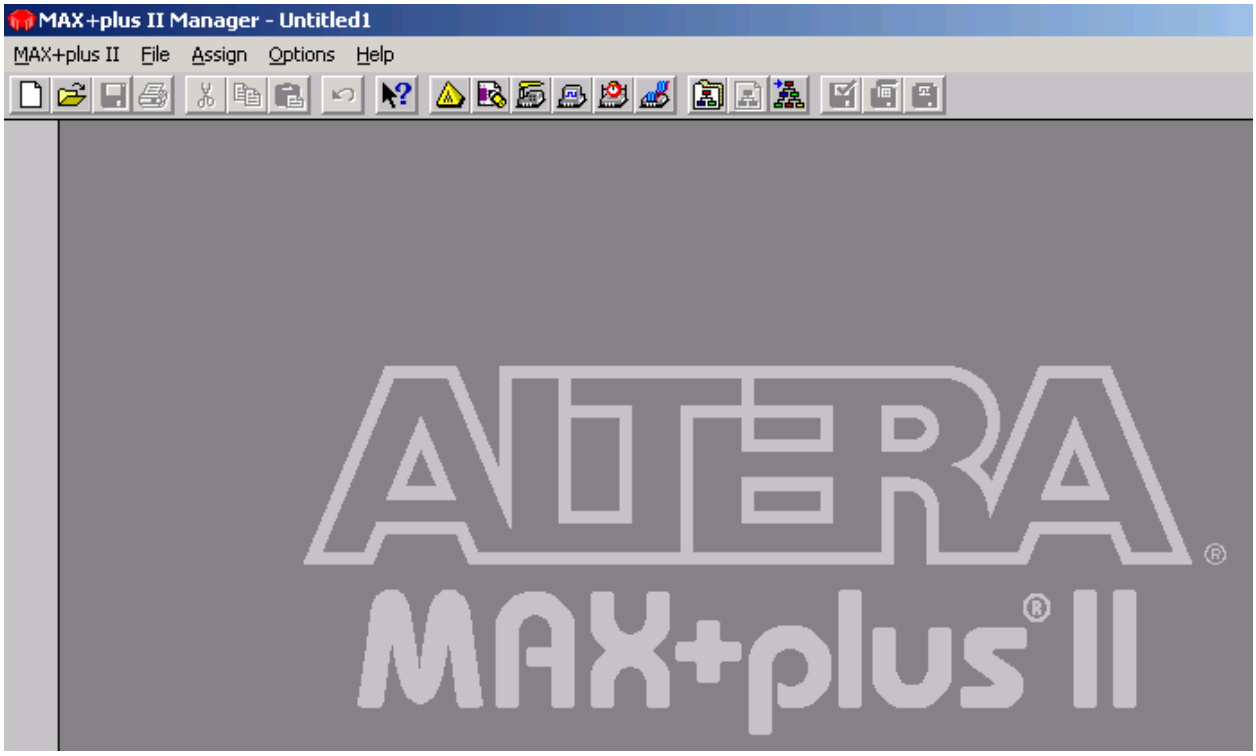

Рисунок 3.1 – Головне вікно САПР MAX + PLUS II BASELANE

Для створення графічного проекту (приклад показано на рисунку 3.2) можна використовувати бібліотеки примітивів, макрофункцій і параметризованних мегафункцій. Примітиви включають великий набір

основних логічних елементів, тригерів, елементів входу та виходу (INPUT, OUTPUT, BIDIR), а також допоміжні елементи: GND (логічний нуль); VCC (логічна одиниця). Макрофункції включають головним чином еквіваленти логічних мікросхем стандартної 74-х серії. Параметризовані мегафункції дозволяють реалізовувати багатовходові та багаторозрядні елементи цифрової схемотехніки (логіку, регістри, мультиплексори тощо), вводячи ряд параметрів у спеціально виділених областях умовних графічних позначень цих елементів.

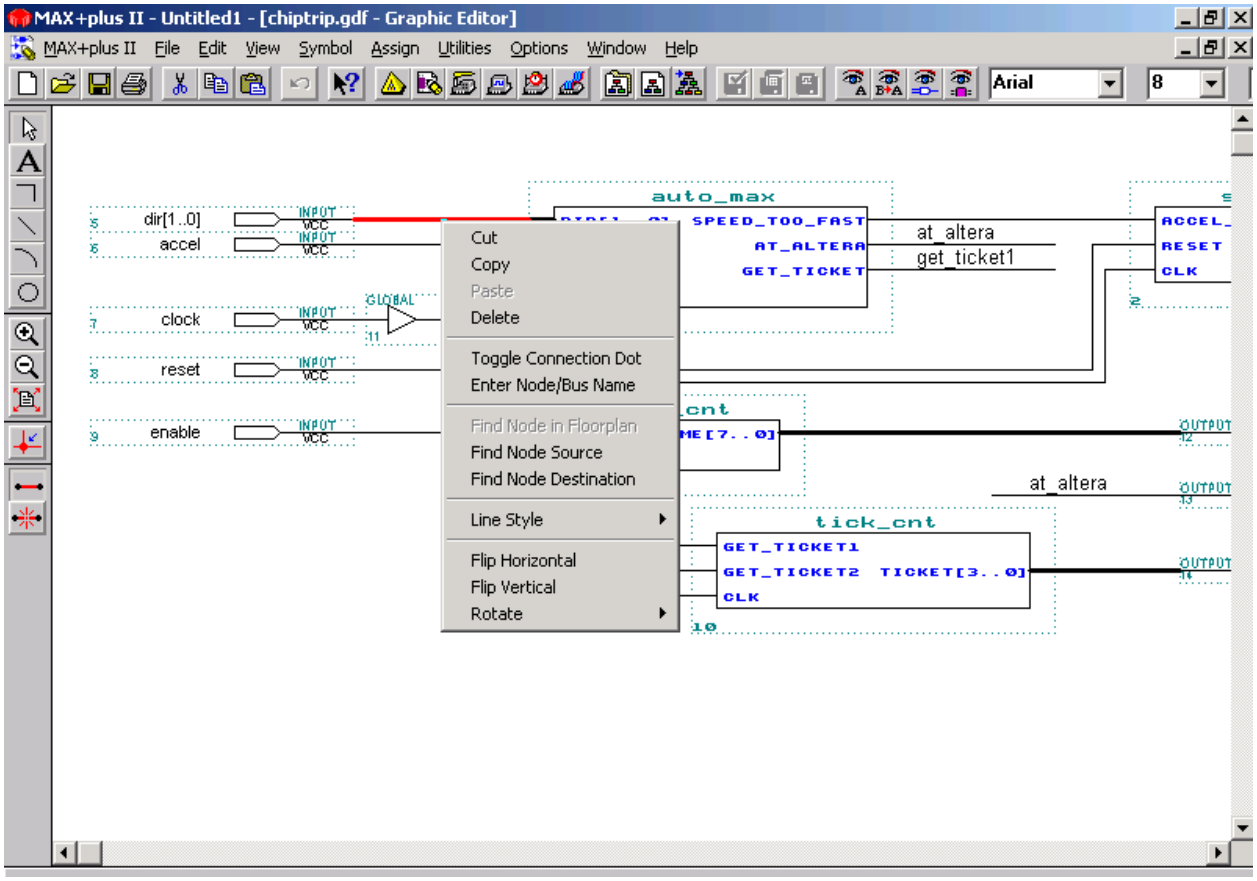

Рисунок 3.2 – Приклад створення проекту у графічному редакторі MAX + PLUS II «Graphic Editor»

На рисунках 3.3-3.5 показано приклади виконання етапів компіляції, часового аналізу та програмування кристалу ПЛІС.

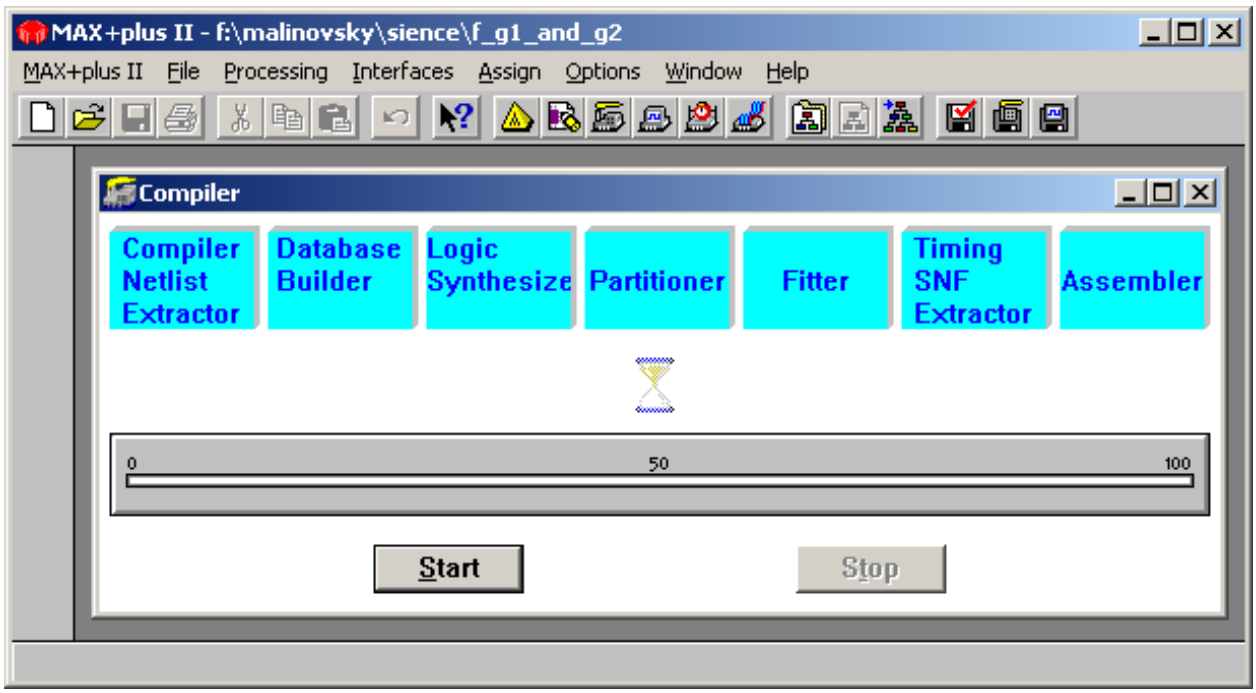

Рисунок 3.3 – Робота MAX + PLUS II Compiler Netlist Extractor

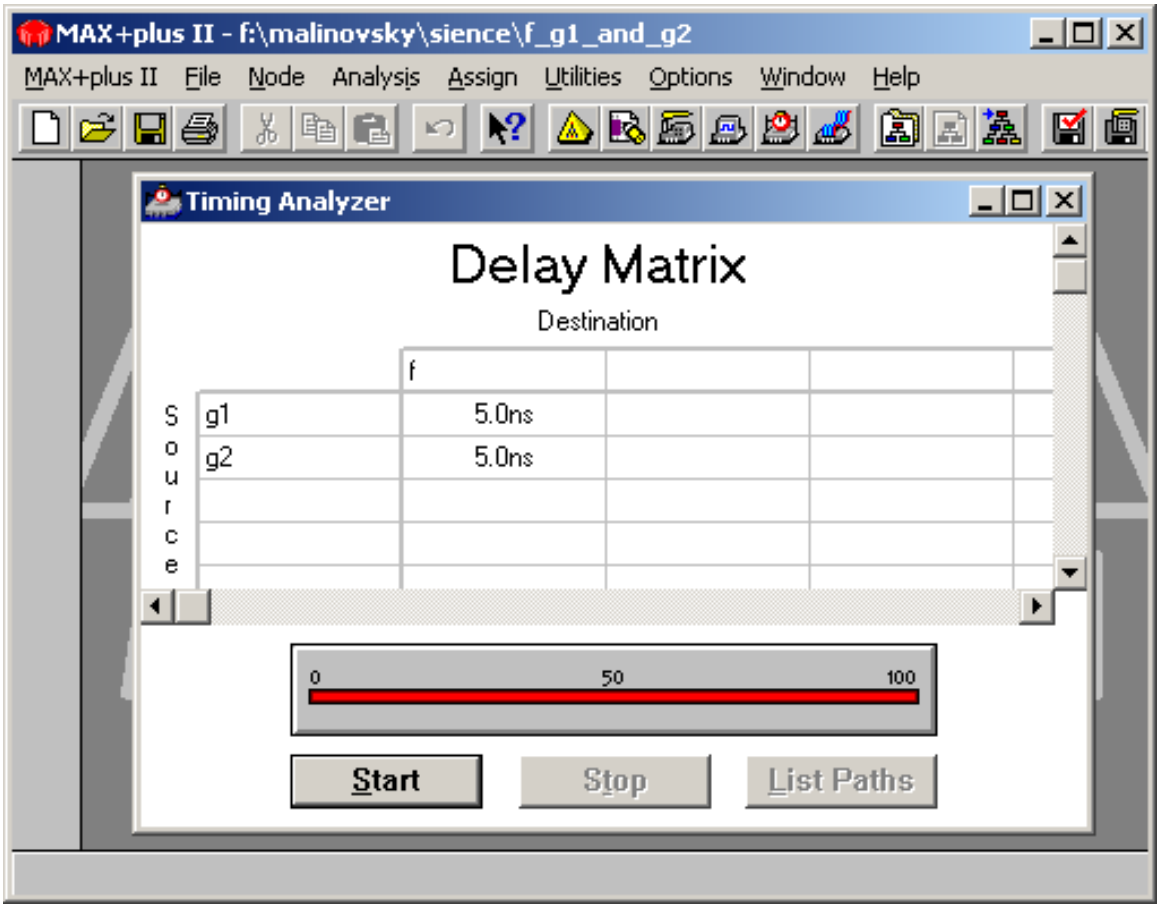

Рисунок 3.4 – Етап виконання часового аналізу в САПР MAX + PLUS II

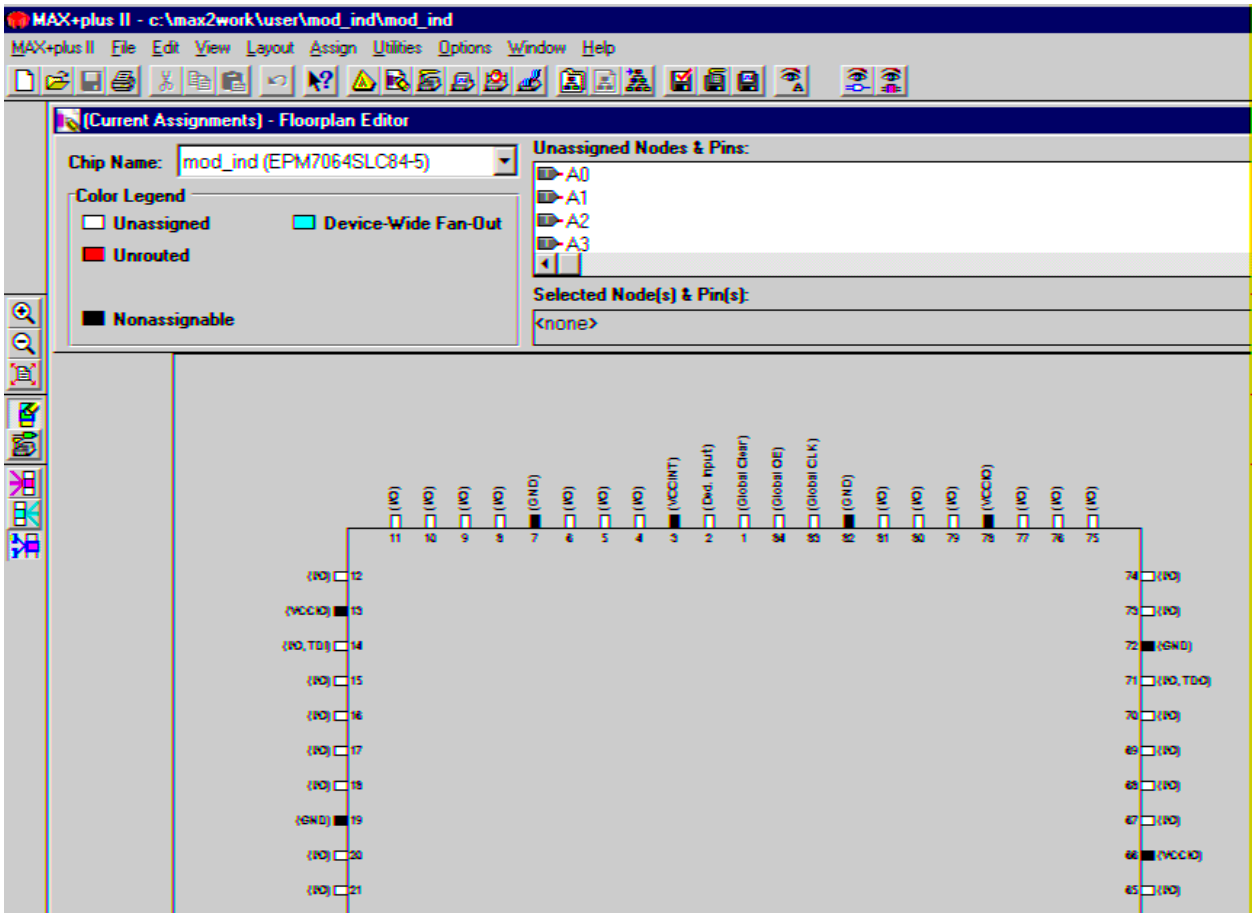

Рисунок 3.5 – Етап виконання програмування кристалу ПЛІС

Усі п'ять редакторів MAX + PLUS II і три редактори створення дизайну (графічний, текстовий та сигнальний) мають загальні функції, такі як, наприклад, створення, збереження та відкриття файлу. Крім того, програми редактора MAX PLUS II мають наступні спільні функції: створення файлів символів та файлів з прототипами функцій (Include файли); пошук вузлів; траверс ієрархічного дерева; спливаючі вікна меню, що залежить від контексту; тимчасовий аналіз; пошук та заміна фрагментів тексту; скасування останнього кроку редагування, його повернення, вирізка, копіювання, вставку та видалення вибраних фрагментів, обмін фрагментами між додатками MAX + PLUS II або додатками Windows; друк.

За допомогою сигнального редактора можна легко перетворювати часові діаграми сигналів повністю або частково, створюючи та редагуючи

вузли та групи. Простими командами можна створювати файл таблиці символів ASCII (.tbl) або імпортувати тестовий файл векторів у форматі ASCII (.vec) для створення файлів тестованих каналів SCF та сигнального дизайну WDF. Можна також зберегти файл WDF як SCF для проведення тестування або перетворити SCF на WDF для використання його як файл проекту.

Для полегшення тестування можна зробити накладення будь-яких виходів у поточному файлі або накласти другий файл сигнального редактора для порівняння сигналів його вузлів та груп з відповідними сигналами поточного файлу. Після виконання всіх призначень та завдання проекту відбувається його компіляція. Спочатку компілятор «витягує» інформацію про ієрархічні зв'язки між файлами проекту та перевіряє проект на прості помилки введення опису проекту.

Компілятор застосовує різноманітні способи підвищення ефективності проекту та мінімізації, використання ресурсів кристалу. Якщо проект занадто великий, щоб бути реалізованим в одному кристалі, компілятор може автоматично розбити його на частини для реалізації в декількох кристалів того ж сімейства, при цьому мінімізується кількість з'єднань між ними. У файлі звіту (.rpt) відображається, як проект буде реалізований в одному або кількох кристалах ПЛІС.

3.2 САПР сімейства Quartus

Програмне забезпечення Quartus II фірми Altera представляє повне мультиплатформне середовище проектування, що включає все етапи проектування цифрового пристрою на ПЛІС за допомогою різних мов опису цифрових пристроїв. Програмне забезпечення Quartus II включає засоби для всіх етапів проектування із застосуванням ПЛІС як FPGA, і CPLD структур.

САПР Quartus II містить у своєму складі:

- різні засоби розробки цифрових пристроїв (графічний, текстовий,

схемний редактори тощо);

- компілятор та редактор для розміщення розробленої схеми на логіку цільового устрою;

- засоби аналізу часових характеристик пристрою (час затримки сигналу між входом та виходом, максимальна тактова частота роботи пристрої тощо);

- редактор часових діаграм для тестування і налагодження розроблюваного пристрою;

- програматор для перенесення конфігурації пристрою з проекту до кристалу ПЛІС.

Додатково Quartus II надає можливість використовувати графічний інтерфейс користувача, EDA або інтерфейс командного рядка в на кожному етапі розробки пристрою. Можна використовувати будь-який інтерфейс для всього процесу проектування або різні налаштування для кожного окремого етапу розробки.

Перерахуємо основні можливості САПР Quartus II.

- різні способи введення поведінкових та структурних описів пристроїв;

- інтегровані засоби допомоги для створення складних проектів MegaWizard® & SOPC;

- підсистема синтезу;

- підсистема розміщення внутрішніх ресурсів та розведення ПЛІС;

- підсистема моделювання;

- підсистема часового аналізу та аналізу споживаної енергії;

- підсистема програмування ПЛІС;

- підсистема оптимізації швидкодії проекту – LogicLock™;

- підсистема підтримки інтеграції з іншими засобами автоматизації проектування – NativeLink®;

- система проектування блоків цифрової обробки сигналів – DSP Builder;

- інтегровані засоби розробки програмного забезпечення для вбудованих мікроЕОМ;

- підтримка використання IP-модулів;

- вбудовані засоби налагодження ПЛІС у складі системи SignalTap® II & SignalProbe™;

- підтримка операційних систем Widows, Solaris та Linux;

- підтримка різних схем ліцензування (nodelocked, network).

Особливості САПР Quartus II.

Засіб розробки Quartus II – це наступний крок від САПР MAX + PLUS II у проектуванні пристроїв з високим ступенем інтеграції, включаючи розробку закінчених систем на одному програмованому кристалі (System-on-a-programmable-chip (SOPC). Quartus II поєднує у собі проектування, синтез, розміщення елементів, трасування з'єднань та верифікацію, зв'язок із системами проектування інших виробників. Розробка систем на кристалі вимагає від розробників ефективної командної роботи, тому зміни в одній частині проекту повинні мати мінімальний вплив на інших членів команди.

LogicLock – це нова блочна методологія проектування, доступна виключно у програмному забезпеченні Quartus II. Quartus II спільно з LogicLock – єдине програмне забезпечення для розробки пристроїв на основі програмованої логіки, яке включає блочну методологію проектування як стандартну функцію Це допомагає збільшити ефективність роботи розробників, знизити час проектування та верифікації. LogicLock дозволяє проектувати та перевіряти кожен модуль окремо. Розробники можуть об'єднувати готові модулі у проект верхнього рівня, зберігаючи продуктивність кожного модуля у процесі об'єднання. LogicLock знижує час розробки та верифікації, оскільки кожен модуль оптимізується лише один раз.

NativeLink – дозволяє здійснювати зв'язок між засобом розробки Quartus II та програмним забезпеченням інших виробників. NativeLink дозволяє засобам синтезу сторонніх виробників перетворювати свої

примітиви безпосередньо до примітивів пристроїв Altera. Пряме перетворення скорочує час компіляції та звільняє від використання додаткових бібліотек трансляцій перетворень, що можуть обмежити продуктивність, досягнуту засобами проектування сторонніх виробників. Процес розробки NativeLink дозволяє розробникам використовувати Quartus II для розміщення елементів, а засоби проектування інших виробників – для оптимізації стратегій синтезу.

Технологія розміщення елементів та трасування з'єднань PowerFit у програмне забезпечення Quartus II використовує часові параметри, задані розробником, для оптимального складання схеми та розміщення логічні елементи. Інтелектуальний алгоритм трасування за тимчасовими параметрам у програмному забезпеченні Quartus II приділяє першорядне увагу з'єднань, критичних до часових параметрів. Критичні до тимчасовим параметрам з'єднання оптимізуються у першу чергу, зменшення затримок та досягнення максимальної продуктивності (f_{MAX}). Ця технологія розміщення елементів та трасування з'єднань допомагає користувачам програмного забезпечення Quartus II досягти максимальною продуктивності і має найменший час компіляції проекту серед таких засобів розробки.

Верифікація. Перевірка або верифікація проекту може виявитися самої тривалою стадією в процесі розробки високопродуктивних систем на кристалі Однак, використовуючи Quartus II, можна скоротити час верифікації, оскільки це програмне забезпечення володіє набором власних засобів верифікації, інтегрованих з останніми засобами верифікації сторонніх фірм.

Аналіз. Altera розробила два методи, щоб допомогти розробникам проаналізувати стан внутрішніх точок і входів/виходів пристрою. Це налагоджувальний засіб SignalProbe та логічний аналізатор SignalTap. Технології SignalTap та SignalProbe можуть працювати разом із засобами синтезу сторонніх виробників і вимагають внесення змін до вихідного файлу проекту HDL.

Технологія апаратної налагодження SignalProbe дозволяє користувачам послідовно з'єднувати внутрішні точки пристрою з вільними зарезервованими виводами для аналізу за допомогою осцилографа або логічного аналізатора.

Для багатьох розробників, які використовують корпуси BGA з великим кількістю входів/виходів, верифікація системного рівня займає дуже багато часу та іноді дуже утруднена. Логічний аналізатор SignalTap виробляє верифікацію, з допомогою інтеграції функціональності логічного аналізатора у програмному забезпеченні. SignalTap дозволяє розробникам зібрати дані з будь-яких внутрішніх точок та входів/виходів пристрої у режимі реального часу під час роботи системи. Quartus II вставляє у проект мегафункцію, що містить логічний аналізатор. Дані збираються та зберігаються в блоках вбудованої пам'яті пристрою та направляються у програмне забезпечення Quartus II через завантажувальний кабель. Розробники можуть подати внутрішні сигнали на виводи пристрою для подальшого моніторингу. Логічний аналізатор SignalTap дозволяє суттєво знизити час верифікації, що дозволяє у більш короткі термін випускати нові продукти.

Програмне забезпечення Quartus II включає технологію PowerGauge – інтегрований засіб аналізу енергоспоживання. Засіб аналізу PowerGauge використовує файли, створені у процесі моделювання для того, щоб зв'язати оцінку споживання енергії із заданими параметрами пристрою. Використовуючи симулятор Quartus II або симулятори сторонніх виробників, інтегрований аналізатор енергоспоживання дозволяє споживачам Altera встановити та оптимізувати споживання енергії на більш ранній стадії процесу розробки.

Розглянемо послідовність етапів проектування в САПР Quartus.

Введення проекту (Design Entry) – бажана схема проекту задається або графічним способом, або з використанням мов опису апаратних засобів, таких як Verilog HDL, VHDL тощо.

Синтез (Synthesis) – проект, що вводиться, синтезується в схему, яка

складається з логічних елементів та логічних блоків кристалу ПЛІС.

Функціональне моделювання (Functional Simulation) – синтезована схема тестується на предмет коректності функціонування вбудованим симулятором, котрий моделює залежність стану (вибраних розробником) сигналів схеми від часу. Це моделювання не враховує тимчасових затримок сигналів (так звані «логічні перегони») між логічними елементами/логічними блоками мікросхеми ПЛІС.

Трасування (Fitting) – інструмент трасування САПР обчислює оптимальне розміщення та з’єднання логічних елементів та логічних блоків, визначених у списку з’єднань (netlist) реальної мікросхеми ПЛІС. Трасувальник також вибирає лінії або «маршрут руху» у чіпі для реалізації необхідних зв’язків між заданими елементами і блоками.

Часовий аналіз (Timing Analysis) – аналізує затримки розповсюдження сигналів вздовж різних шляхів в логічній схемі для обчислення наявності або відсутності «логічних перегонів», щоб сигнали з різних логічного блоку приходили одночасно в той чи інший кінцевий вузол схеми

Часове моделювання (Timing Simulation) – схема тестується для перевірок функціональності та часових обмежень, але на відміну від функціонального моделювання, тут для виявлення наявності або відсутності логічних перегонів враховуються реальні затримки обраних сигналів.

Програмування і конфігурація (Programming and Configuration) – розроблена схема розміщується на кристалі ПЛІС шляхом програмування електронних зв’язків між конфігурованими логічними елементами/логічними блоками. Цей етап реалізується шляхом передачі конфігураційного файлу з комп’ютера або в мікросхему ПЛІС, або додаткову (не вбудовану в ПЛІС) пам'ять, якщо така є у комплекті налагодження.

Приклади етапів створення проекту в САПР Quartus показано на рисунках 3.6-3.8.

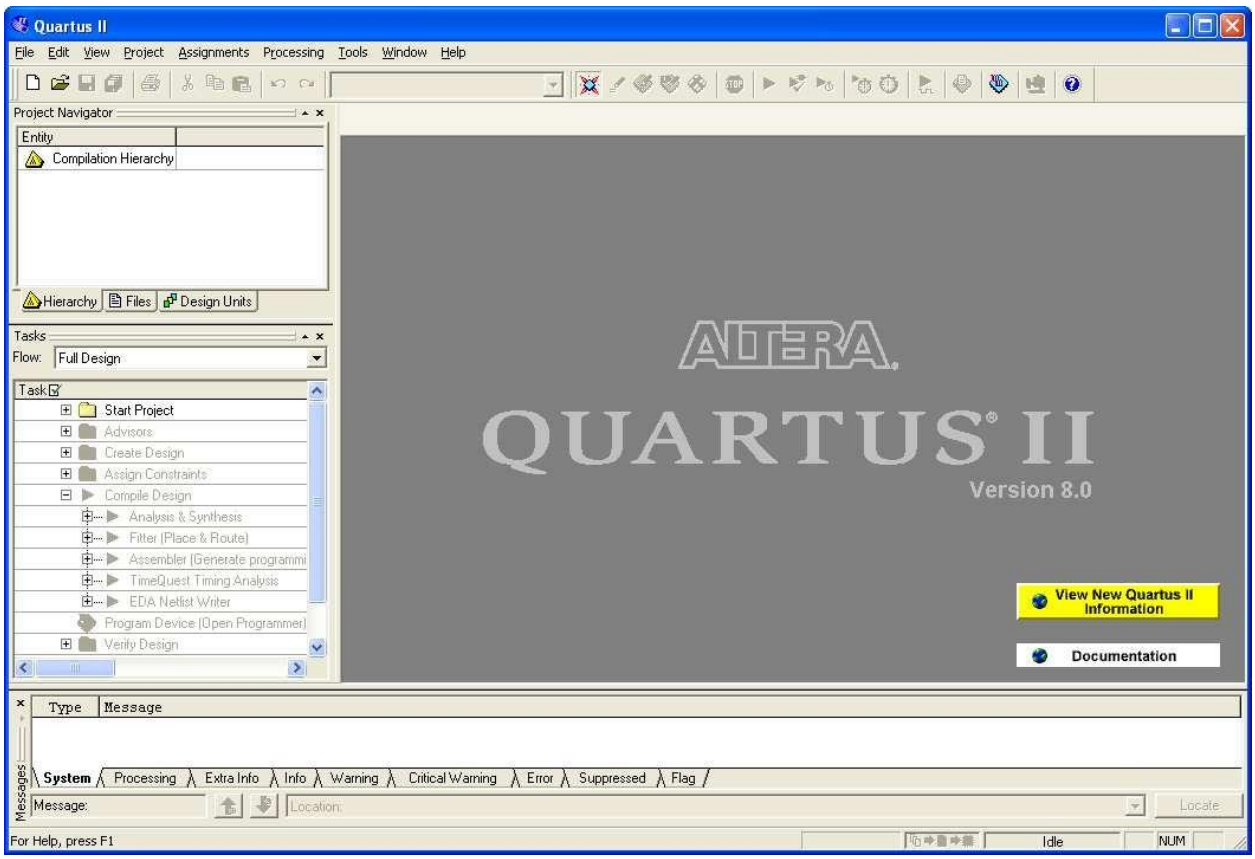

Рисунок 3.6 – Головне вікно САПР Quartus II

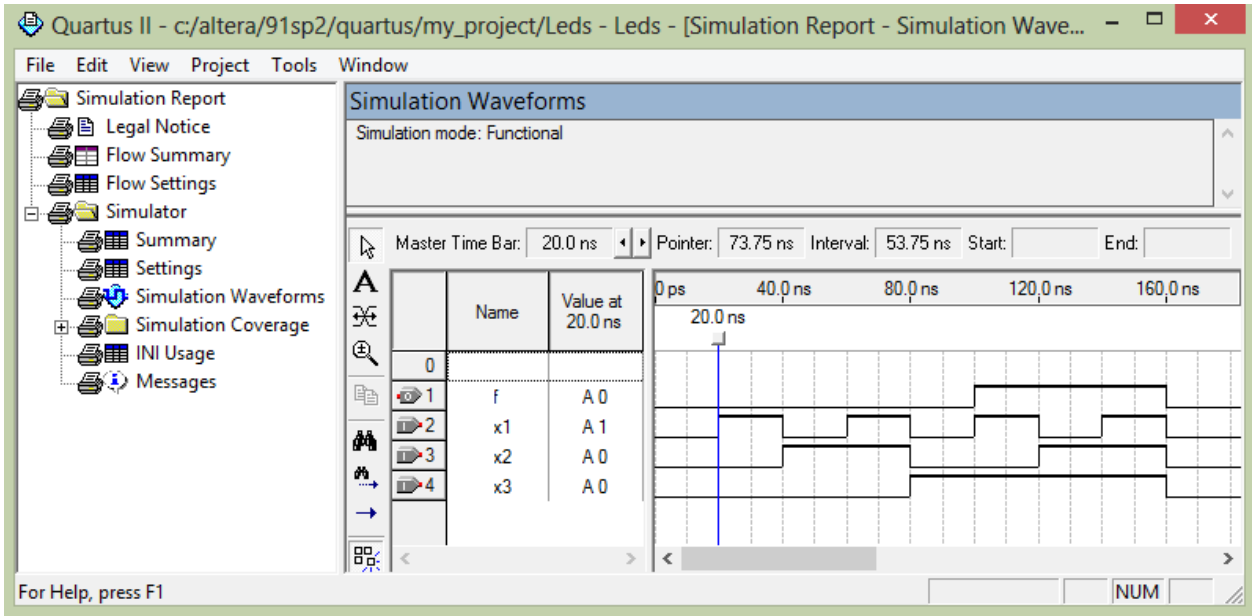

Рисунок 3.7 – Результати функціонального моделювання в середовищі САПР Quartus II

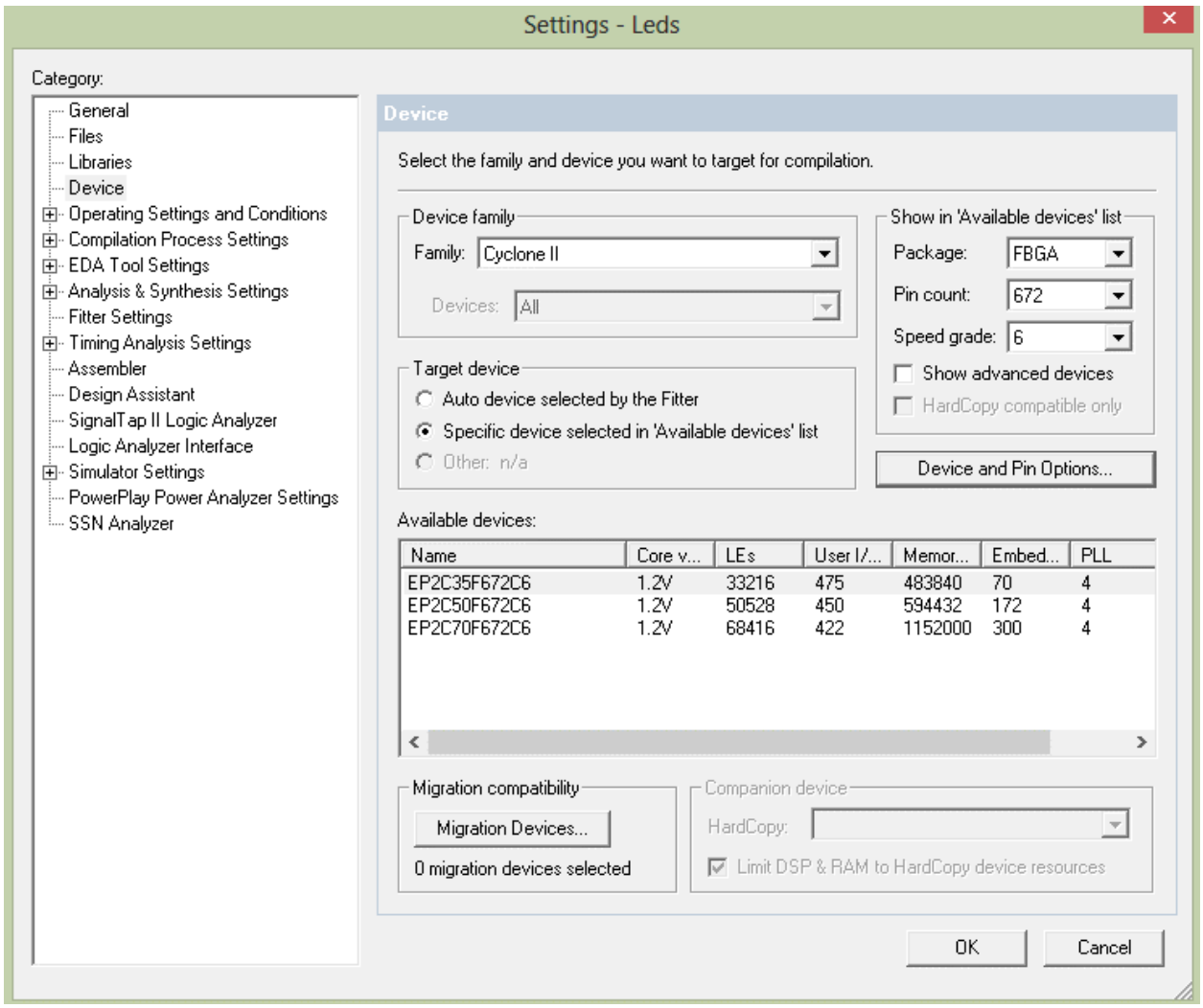

Рисунок 3.8 – Програмування кристалу ПЛІС в САПР Quartus II

У 2015 році компанію Altera придбав всесвітній гігант Intel. Поступово всі продукти Altera, як апаратні, так і програмні, змінили свою назву. Зокрема, програмне забезпечення для дизайну FPGA-систем Altera Quartus стало іменуватися Intel Quartus Prime.

Компанія Intel, є світовим лідером у виробництві FPGA. Середовище САПР дозволяє проектувати логіку роботи мікросхем схемотехнічно та мовами програмування AHDL, VHDL, Verilog та інших. Середовище програмування Intel Quartus Prime також дозволяє виробляти симуляцію проектів, завантажувати скомпільований образ FPGA в мікросхему, вести внутрішньосхемне налагодження проектів та багато іншого. Альтера і потім Інтел, протягом багатьох років удосконалювала середу Quartus. Попередня

версія цієї програми раніше називалася Altera Quartus II.

Існують безкоштовні, але функціональні версії САПР Quartus II Web Edition або остання Quartus Prime Lite. Є й платні передплати. Однак, навіть для професійної роботи найчастіше вистачає безкоштовних версій Quartus. Безперечно, відмінності між Quartus II і Quartus Prime є. Насамперед необхідно звертати увагу, що для різних серій ПЛІС може знадобитися різна САПР.

Варіанти САПР Quartus Prime і підтримувані ними сімейства кристалів ПЛІС показано у таблиці 3.1.

Таблиця 3.1 – Підтримка САПР Quartus Prime сімейств FPGA

Пристрої	Pro	Standard	Lite
Stratix IV, V		+	
Intel Stratix 10	+		
Arria II			+
Arria II, V		+	
Intel Arria 10	+	+	
Cyclone IV, V		+	+
Intel Cyclone 10 LP		+	+
Intel Cyclone 10 GX	+		
Intel MAX series		+	+

Intel Quartus Prime містить все, що необхідно для розробки дизайну систем на базі Intel FPGA, SoC і Complex Programmable Logic Device (CPLD), починаючи з самих основ і включаючи налагодження взаємодії, оптимізацію, верифікацію та моделювання. В даний час існує три варіанти поставки Quartus Prime:

- Intel Quartus Prime Pro Edition призначена для роботи з сучасними рішеннями FPGA і SoC останнього покоління, такими як Intel Stratix 10, Intel Arria 10, Intel Cyclone 10 GX;

- The Intel Quartus Prime Standard Edition включає повну підтримку пристроїв попередніх поколінь, а також сімейства Intel Cyclone 10 LP;

- The Intel Quartus Prime Lite Edition є інструментом для роботи з сімействами масового сегмента, його можна скачати безкоштовно без додаткового ліцензування.

4 РОЗРОБКА HDL-МОДЕЛІ ПЕРСПЕКТИВНОГО ЛКА ПД

Виходячи з тих змін, що зазнала структура класичного ЛКА ПД у частині реалізації елементів, що реалізують функціонал програмованих користувачем таймерів, пропонується внести наступні модифікації до його, що наводяться нижче.

Найбільш важливою зміною є додавання елементів до HDL-моделі, що призначені для реалізації додаткових структурних блоків ПЛІС-контролера паралельної дії (рисунок 1.3). Такими блоками є блок пам’яті внутрішніх змінних – БПВЗ та блок внутрішніх таймерів – БВТ. Це знайшло своє відображення у появі в HDL-моделі додаткових функціональних блоків ***bpvz:_bpvz*** та ***bvt:_bvt*** (рисунок 4.1).

Окрім цього зміни відбулись у появі додаткових входів і виходів, або призначенні внутрішніх сигналів моделі у наступних функціональних блоках: блоці логічного керування ***blk:_blk***, блоці пам’яті переходів – ***bpp:_bpp***, блоці пам’яті команд ***bpk:_bpk***.

Розглянемо взаємодію функціональних блоків пропонованої HDL-моделі із зазначенням функцій кожного з них.

Функціональний блок *bps:_bps* – БПС. Виконує функції, що аналогічні до функцій блоку пам’яті станів класичного ЛКА ПД, тобто до нього записані «дозволені» комбінації станів детермінованих входів. Блок ***bps:_bps*** так само зберігає *k*-розрядні вектори двобітних даних *vec_Ai[1..k][1..2], що визначаються* *q*-розрядною вхідною адресою *adr[q..1]*. Функціонування блоку ***bps:_bps*** аналогічне моделі класичного ЛКА ПД.

Функціональний блок *bpk:_bpk* – БПК. На відміну від розглянутої у другому розділі роботи моделі, відбулись деякі зміни. Вихід, що призначений для реалізації сигналу «кінець підпрограми» отримав нове позначення *endsp1*, оскільки перехід до наступної підпрограми може

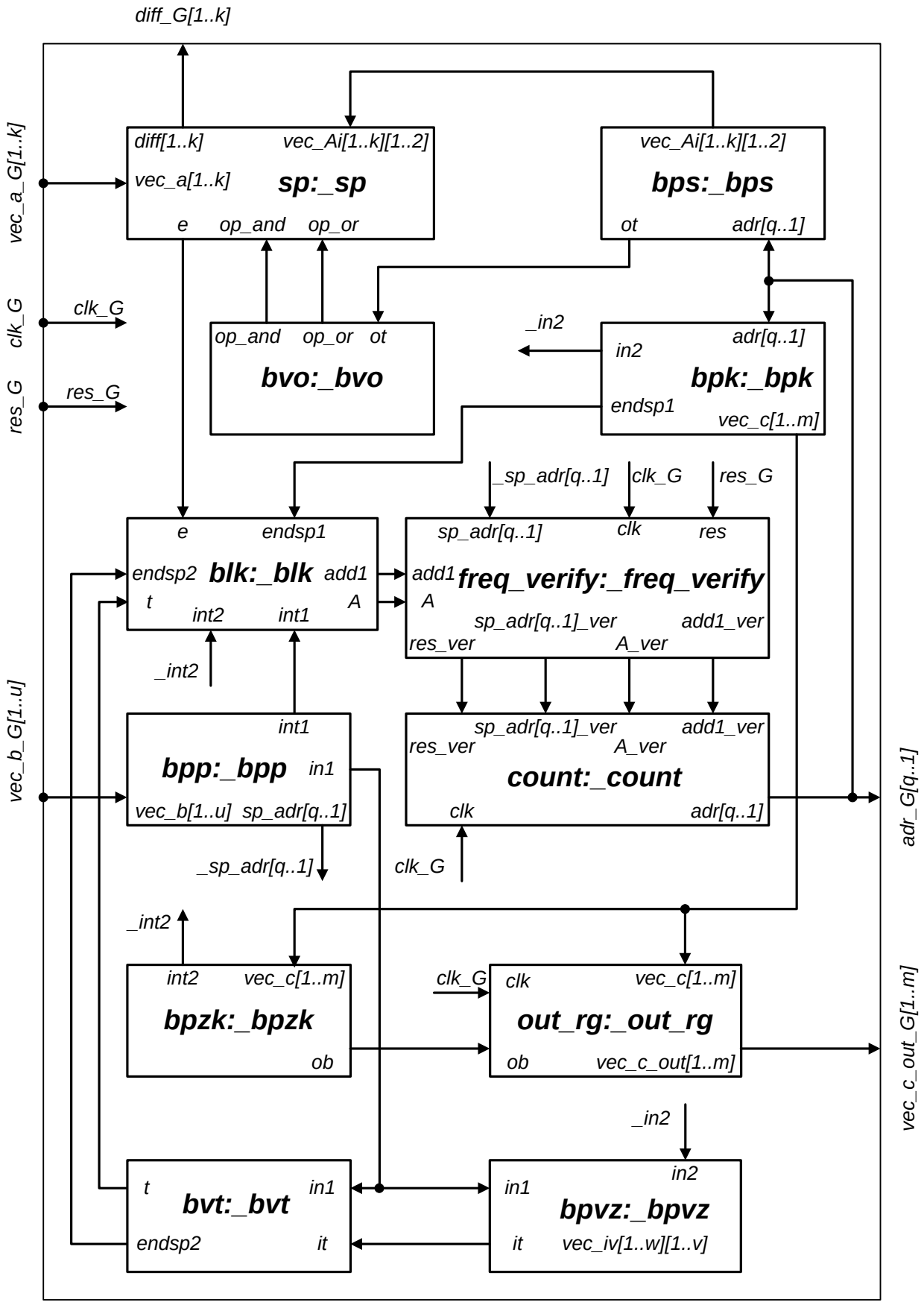

Рисунок 4.1 – HDL-модель пропонованого ЛКА ПД

відбуватись (у загальному випадку, тобто без наявності аварійних ситуацій) не тільки по закінченню відпрацювання попередньої підпрограми, але і за наявністю відповідного сигналу від внутрішнього програмованого таймера. також до *i*-го рядку блоку пам'яті команд може бути записано команду ініціатор відліку таймера, це знайшло своє відображення у HDL-моделі у вигляді появи додаткового виходу *in2* у функціональному блоці ***bpk:_bpk***.

Функціональний блок *bpp:_bpp* – блок БПП. Повністю зберіг своє функціональне призначення і структуру. Блок призначений для зберігання *q*-розрядних початкових адрес підпрограм – *sp_adr[q..1]*, що визначаються *u*-розрядним вхідним вектором *vec_b[1..u]*.

Так само не змінив свого функціоналу **блок *bpzk:_bpzk*** – блок пам'яті заборонених комбінацій. Блок виконує функції блокування видачі вектора *vec_c[1..m]* на виконавчі елементи керованого агрегату, у випадку якщо вектор містить заборонену їх комбінацію.

Функціональний блок *sp:_sp* – призначений для реалізації блоку СП структури ЛКА ПД. Оскільки робота внутрішніх програмованих таймерів не залежить від процедури порівняння сигналів детермінованих входів з їх очікуваними значеннями, то блок ***sp:_sp*** не змінився у порівнянні з HDL-моделлю класичного ЛКА ПД. Нагадаємо, що основним призначенням цього функціонального блоку є формування сигналу *e* у випадку порозрядного співпадіння сигналів векторів *vec_a[1..k]* і *vec_Ai[1..k][1..2]*, або при співпадінні сигналів хоча б в одному із розрядів вхідних векторів, у залежності від типу обраної логічної операції Тип цієї операції, як і раніше визначається функціональним блоком ***bvo:_bvo***, що також присутній в пропонованій HDL-моделі.

Функціональний блок *count:_count* – блок призначений для формування роботи лічильника адреси (ЛА). Цей блок забезпечує адресацію блоків пам'яті станів та команд, тобто визначає рядок поточної підпрограми. Оскільки внутрішні таймери безпосередньо не визначають адресу. то цей функціональний блок повністю збігається з аналогічним блоком HDL-моделі

класичного ЛКА ПД.

Функціональний блок *blk:_blk*, або блок логічного керування. Як було вказано в розділі 2, цей блок має зазнати найбільших змін, у порівнянні з класичною HDL-моделлю, але для спрощення опису роботи пропонованого ЛКА ПД, функції БЛК, що визначають роботу з таймерами, було штучно винесено до окремого блоку – БВТ, опис якого буде наведено нижче. Зараз зупинимось більш детально лише на змінах, що зазнав ***блок blk:_blk*** у розрізі загальної логіки роботи ЛКА ПД. Логічні входи *e* (еквівалентність), *int1* і *int2* (виходи формування сигналів аварійного переривання 1 і 2) повністю зберігли свою функціональність і призначення. Так само і виходи керування лічильником адреси *A* (паралельне завантаження адреси) та *add1* (додавання одиниці) мають призначення і логіку роботи аналогічні класичному ЛКА ПД. змінив свою назву логічний вхід *endsp* (кінець підпрограми), що тепер називається *endsp1*, це пов'язано з тим, що перехід до відпрацювання іншої підпрограми тепер може бути ініційований не тільки блоком пам'яті переходів, але і внутрішнім програмованим таймером.

Функціональний блок *out_rg:_out_rg*. Вихідний регістр не приймає участі у формуванні логіки роботи ЛКА ПД, а призначений лише для тимчасового зберігання вихідного вектора команд керування *vec_c[1..m]* і, у випадку наявності «забороненої» комбінації, її блокування. і не допущення видачі на виконавчі механізми. Отже у пропонованій моделі функціональний блок ***out_rg:_out_rg*** не зазнав жодних змін.

Функціональний блок багатократного контролю *freq_verify:_freq_verify*. Оскільки цей блок призначений для багатократної перевірки достовірності сигналів у яких безпосередньо не приймають участі внутрішні таймері, то цей блок також не отримав жодних змін. Він, як і раніше призначений для роботи з загальним сигналом синхронізації кристалу ПЛІС *clk_G* та сигналів, що визначають внутрішню логіку роботи ЛКА ПД *res, A, add1*. Внутрішня структура блоку ***freq_verify:_freq_verify*** відповідно також не змінилась і складається з компонентів: ***freq_verify_reg:reg1***,

freq_verify_reg2:reg2, ***freq_verify_comparator:comparator***, ***freq_verify_counter: counter***, ***tr (tr1-tr[q+3])***, ***tro (tro1, tro2)*** та ***troA***.

Тепер розглянемо призначення і функціональність додаткових блоків, що були уведені до пропонованої HDL-моделі і призначення яких – безпосередня реалізація функції програмованих користувачем таймерів.

Функціональний блок *bvt:_bvt* – блок внутрішніх таймерів. Як вже було показано раніше, цей блок дозволяє інтегрувати логіку роботи програмованих користувачем таймерів до загальної логіки роботи ЛКА ПД у цілому. Блок має вхід *it* – вхід внутрішнього таймера, поява логічної одиниці на виході якого, призведе до формування, одного з двох вихідних сигналів або *t,* або *ends2,* у залежності від того, який саме функціональний блок (***bpk:_bpk***, або ***bpp:_bpp***) був ініціатором формування відліку часу внутрішнім таймером. Інформацію про те, який саме блок був ініціатором, блок ***bvt:_bvt*** отримує з входу *in1*, якщо на ньому зафіксовано значення логічної одиниці, то це означає, що ініціатором є блок пам’яті переходів ***bpp:_bpp*** і потрібно формувати ознаку кінці підпрограми, тобто сигналу *ends2*. У випадку наявності сигналу логічного нуля на вході *in1* блок ***bvt:_bvt*** приймає рішення, що ініціатором був блок пам’яті команд ***bpk:_bpk***, що призведе до формування сигналу *t* і, як наслідок переходу до виконання наступного рядка поточної підпрограми.

Функціональний блок *bpvz:_bpvz* – блок пам’яті внутрішніх змінних. Власне це і є блок, який призначений для зберігання чисельних значень відліків часу, що мають бути сформовані програмованими таймерами. У блоці передбачено два входи *in1* та *in2*. Сигнали на цих входах дозволяють диференціювати джерело запуску процедури відліку внутрішніх програмованих таймерів, або ***bpk:_bpk***, або ***bpp:_bpp***. Сигнал логічної одиниці на виході *it* функціонального блоку ***bpvz:_bpvz*** буде сформовано у момент, коли відлік проміжку часу таймером, буде завершено. Слід зазначити, що функціонального блоку ***bpvz:_bpvz*** у подальшому може бути використаний для реалізації не тільки таймерів, а і, наприклад, лічильників,

або зберігання деяких констант, тощо.

Таким чином, представлена HDL-модель логічного керуючого автомата паралельної дії дозволяє виконати перехід від абстрактної структури, до практичної реалізації ПЛІС-контролера паралельної дії засобами САПР MAX+PLUS II або Quartus.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено аналіз керуючих структур з паралельною архітектурою, розглянуто структуру класичного ПЛІС-контролера паралельної дії, досліджено і визначено особливості ПЛІС-контролера з розширеними функціональними можливостями.

Розглянуто загальні характеристики та особливості застосування сучасних кристалів ПЛІС. Проведено аналіз HDL-моделі ПЛІС-контролера ПД, визначено можливості і підходи до її вдосконалення та інтеграції рішень для реалізації програмованих користувачем таймерів.

Проведено детальний аналіз інструментальних засобів розробки HDL-моделей керуючих пристроїв з паралельною архітектурою.

Виконано розробку HDL-моделі вдосконаленого логічного керуючого автомата паралельної дії, у складі якого присутні елементі, що реалізують програмовані користувачем таймери.

Результати наукових досліджень кваліфікаційної роботи опубліковано у збірнику наукових праць «Системи управління навігації та зв’язку» випуск 2 (76) 2024 року [6].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

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

16. 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.

17. 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.

18. 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.

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.