Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

перший (бакалаврський)
(освітній ступінь)

ГЮІК. 46741Х.039 ПЗ
(позначення документа)

Вбудований пристрій діагностування кінцевих автоматів
(тема)

Виконав: студент *IV* курсу, групи *КІ-14-7*

спеціальності (напряму підготовки) _____

6.050102 – Комп'ютерна інженерія

_____
(шифр і назва спеціальності, напряму)

_____ *Гога М.В.*
(підпис) (прізвище, ініціали)

Керівник _____ *Шкіль О.С.*
(підпис) (прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2018 р.

Харківський національний університет радіоелектроніки

Факультет_____

Кафедра_____

Рівень вищої освіти____перший (бакалаврський)_____

Спеціальність_____

(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2018 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ (ПРОЕКТ)

Студентові Гога Максиму Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) «Вбудований пристрій діагностування кінцевих автоматів»_____
_____
затверджена наказом по університету від " 25 "_____05_____2018 р. № 608 Ст._____
2. Термін подання студентом роботи (проекту) _____11.06.2018_____
3. Вихідні дані до роботи (проекту)_____
Мова опису апаратури VHDL,_____
САПР Active-HDL_____
САПР XILINX ISE _____
ПЛІС Spartan 3E_____
_____
_____
4. Перелік питань, що потрібно опрацювати у роботі _____
Аналіз предметної галузі та постановка задачі проектування._____
Розробка моделі об'єкта діагностування_____
Розробка HDL-моделі пристрою діагностування _____
Моделювання розробленихої моделі з використанняс інструментальних_____
засобів Active-HDL_____
Синтез апаратної реалізазіі пристрою з використанням САПР XILINX ISE та аналіз апаратурних витрат
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) ___17___слайдів_____

_____

_____

_____

_____

_____

6. Консультанти розділів роботи (проекту)

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

7. Дата видачі завдання 07.05.2018

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Видача теми проекту, узгодження і затвердження	7.05.2018 – 10.05.2018	
2	Аналіз проблемної галузі, постановка задачі, вибір інструментальних засобів	11.05.2018 – 15.05.2015	
3	Розробка методів побудови тестів обходу графа	16.05.2016 – 18.05.2016	
4	Розробка методики застосування інструментів Active-HDL	19.05.2016 – 21.05.2016	
5	Розробка програми TestBench	22.05.2016 – 25.05.2016	
6	Проведення діагностичних експериментів	25.05.2016 – 27.05.2016	
7	Оформлення пояснювальної записки	28.05.2016 – 01.06.2016	
8	Перевірка виконаного проекту керівником,	04.06.2016 – 06.06.2016	
9	Захист проекту	11.06.2016 – 20.06.2016	

Студент _____
(підпис)

Керівник роботи (проекту) _____ Доц. каф. АПОТ Шкіль О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 60 сторінок, 18 рисунків, 3 таблиці, 8 джерел за переліком посилань.

VHDL-МОДЕЛЬ, КІНЦЕВІ АВТОМАТИ, СИНТЕЗ, ПРИСТРОЇ ДІАГНОСТУВАННЯ, АПАРАТНІ ВИТРАТИ

Розглянуті питання апаратного діагностування пристроїв управління станціями газорозподілу. Алгоритм управління, описується граф-схемою алгоритму, яка представляється моделлю графа переходів кінцевого автомата. Для реалізації алгоритму діагностування використовується спосіб обходу усіх дуг графа. Модель керуючого автомата і пристрою діагностування описуються на мові опису апаратури VHDL з наступним синтезом та використанням пристроїв програмованої логіки. Проаналізовані апаратні витрати для різних варіантів реалізації пристрою діагностування.

ABSTRACT

Bachelor’s thesis contains 60 pages, 18 figures, 3 tables, 8 sources according to the list of links.

VHDL-MODEL, FINITE STATE MACHINE, SYNTHESIS, DIAGNOSTIC DEVICE, HARDWARE COSTS

The questions of hardware diagnostics of control devices by gas-distributing stations are considered. The control algorithm is described by the graph-diagram of the algorithmic state machine, which is represented by the model of the state diagram of the finite state machine. To implement the diagnostic algorithm, a method is used to bypass all arcs of the state diagram. The model of the control finite state machine and the diagnostic device are described in the hardware description language of the VHDL, followed by synthesis using programmable logic devices. The hardware costs for various variants of implementation of the diagnostic device are analyzed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ ……..……..……7
ВСТУП ……..…….……..8
1 ВБУДОВАНЕ ДІАГНОСТУВАННЯ АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ ……..…10
1.1 Методи контролю автоматизованих систем керування…….10
1.2 Вбудовані системи діагностування в електроенергетиці ……19
1.3 Процедури діагностування HDL-моделей кінцевих автоматів ……21
2 МОДЕЛЬ ОБ'ЄКТА ДІАГНОСТУВАННЯ ……24
2.1 Автоматична станція газорозподілу ……..……..…24
2.2 Модель керуючого автомата в системі управління АГРС ……...…27
3 МОДЕЛЬ ВБУДОВАНОГО ПРИСТРОЮ ДІАГНОСТУВАННЯ …...…34
3.1 Стратегія діагностування керуючого автомата ……..…34
3.2 VHDL-моделі пристрою діагностування ……...……36
3.2.1 Структура та алгоритм пристрою діагностування …….….36
3.2.2 Бітове кодування алгоритму діагностування ……...…39
3.2.3 Алгоритм діагностування за кодами станів автомата …….……43
3.2.4 Алгоритм діагностування з використанням регістру зсуву …...48
3.2.5 Аналіз витрат апаратури ……54
3.3 Загальна модель керуючого пристрою і діагностування ……..……55
ВИСНОВКИ ……...…58
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ……..……..……60
ДОДАТОК А Графічна частина проекту ……...……61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASIC – application-specific integrated circuit, (інтегральна схема спеціального призначення)

АС – автоматизовані системи

САР – системи автоматичного регулювання

САК – системи автоматичного керування

АГРС – автоматична газорозподільна станція

ПЛІС – програмована логічна інтегральна схема

КП – керуючий пристрій

ПД – пристрій діагностування

ВСТУП

В даний час в електроенергетиці відбуваються суттєві зміни, пов'язані з впровадженням нових технологій автоматизації та енергозбереження. На рівні підприємств і територій використовуються автоматизовані системи керування технологічними процесами, які, в основному, керують інформаційними потоками. У системах енерго- та газопостачання широко використовуються спеціальні локальні системи управління і регулювання, розташовані на віддаленій місцевості, наприклад на трансформаторних підстанціях, газорозподільних вузлах, пунктах обліку енергоспоживання і т.д.

Системи автоматичного регулювання (САР) призначені для підтримки постійної або зміни по заданому закону деякої керованої величини. Системи автоматичного керування (САК) здійснюють сукупний вплив на об'єкт, обраний з безлічі можливих впливів, спрямованих на досягнення певного критерію керування. У загальному випадку, САР і САК можуть будуватися як на основі локальних засобів автоматичного регулювання, так і з застосуванням цифрових систем автоматичного керування. Такі локальні системи, як правило, реалізуються на інтегральних схемах, виготовлених на замовлення або мікроконтролерах.

Важливим завданням при побудові САК є забезпечення надійності їх функціонування, що неможливо без використання автоматичних систем технічного діагностування. При проектуванні сучасних систем технічного діагностування широко використовуються комп'ютерні технології автоматизованого проектування з застосуванням мов опису апаратури і сучасної технологічної бази. Даний підхід дозволить реалізувати систему діагностування віддаленого пункту управління будь-якої енергетичної системи без участі людини і без відключення основної системи управління на тривалий час.

Виходячи з огляду публікацій з питання побудови вбудованих систем діагностування [1, 2, 3] можна зробити висновок, що питання побудови апаратних систем діагностування в локальних системах автоматичного управління виробничими процесами на сучасній технологічній базі потребують подальших досліджень.

В електроенергетиці і газопостачанні досить поширені віддалені пункти управління, які працюють без / або з мінімальною участю людини, а також без використання персональних комп'ютерів. При цьому пристрої управління реалізуються на технологічній базі МК, ASIC або ПЛІС. Незалежно від способу технічної реалізації зазначені системи реалізують оригінальний алгоритм управління, описаний відповідною граф-схемою. При цьому виникає проблема діагностики правильного функціонування керуючого пристрою без використання показань реальних датчиків, тому що їх включення в режим діагностування може порушити процес функціонування критичних систем електроенергетики та газопостачання. Процес діагностування керуючого пристрою має йти в автономному режимі при відключенні систем управління на досить короткий час. Тому актуальною є задача розробки автоматичних апаратних засобів діагностування, які працюють в автономному режимі без участі людини і гарантують задану повноту діагностування.

1 ВБУДОВАНЕ ДІАГНОСТУВАННЯ АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ

1.1 Методи контролю автоматизованих систем керування

Ефективність функціонування автоматизованих систем (АС) в значній мірі залежить від того, наскільки повно, при виборі методів і засобів контролю, були враховані особливості зазначених систем як об'єктів контролю. Без урахування цих особливостей АС практично неможливо встановити для кожної з них роль і місце контролю в управлінні її функціонуванням. Специфіка АС як об'єктів контролю впливає на вибір методів і видів контролю, а також показників якості їх функціонування. Вона знаходить своє відображення в принципах формалізації і змістовного опису процесів контролю, в принципах синтезу функцій працездатності АС [4].

Зі зростом виробництва інтенсивно ростуть потоки інформації, що перероблюються, причому в загальному випадку обсяг інформації зростає, приблизно, пропорційно квадрату кількості виробленої продукції. Вимоги до якості і швидкості передачі і переробки інформації все більше посилюються. Збільшується загальний обсяг виробництва при зростаючих розмірах і спеціалізації підприємств. Для підприємств, що входять в замкнутий виробничий цикл, характерні значне територіальне розосередження та ієрархічна структура побудови. Зростає роль централізації управління та узгодженості роботи систем різного рівня.

Якщо з яких-небудь причин хоча б одна з таких систем не виконує покладених на неї функцій – це призводить до відхилення виробничого процесу від оптимального протікання або ж взагалі до неприпустимого його порушення. Для запобігання втрат і несприятливих наслідків необхідно своєчасно отримувати інформацію про правильність виконання функцій системою.

Автоматичні системи можуть виконувати покладені на них функції тільки в тому випадку, якщо вони мають властивість збереження працездатності протягом заданого інтервалу часу в певних умовах експлуатації, тобто якщо мають необхідну безвідмовність.

Для підвищення надійності автоматичних систем широко застосовують різні методи резервування, проте можливості резервування обмежені. Це обумовлено тим, що значне збільшення середнього часу між відмовами пристрою, навіть в граничному випадку, може бути досягнуто тільки при практично не реалізованій загальній кількості його елементів. До того ж, чим більше кратність резервування, тим менше її відносна ефективність.

Безвідмовність дискретних пристроїв може бути підвищена введенням структурної надлишковості, при якій виходи пристроїв або систем, що знаходяться в резервованому з'єднанні, об'єднуються логічним органом відновлення (мажоритарним елементом). Таке резервування може бути реалізовано і на більш низькому рівні. Для виявлення відмови в будь-якому з каналів досить дублювання, а відновлення інформації можливо при трьох і більше паралельно працюючих каналах. Таке резервування дуже ефективне, особливо при боротьбі зі збоями, проте вимагає великої надлишковості. Його ефективність значно знижується при виникненні відмови.

Для підвищення безвідмовності роботи систем може бути, також, використана інформаційна надлишковість. У цьому випадку надлишкова інформація призначена як для виявлення, так і для виправлення викривлень в робочій інформації.

Слід мати на увазі, що інформаційна надлишковість неминуче призводить до структурної надлишковості, що ускладнює систему і знижує її безвідмовність. Інформаційна надлишковість ефективна тільки при ліквідації наслідків збоїв. Що ж стосується зниження впливу відмов, то можливості її дуже обмежені.

З розглянутого легко зробити висновок про те, що можливості забезпечення необхідної ефективності функціонування систем, тільки за

допомогою реалізації прямих методів підвищення безвідмовності апаратури, обмежені. Поряд із застосуванням цих методів необхідна ефективність може бути досягнута за допомогою своєчасного відновлення відмовлених пристроїв і викривленої, циркулюючої в них, інформації. Для реалізації відновлення необхідно мати відомості про стан апаратури системи, а також про якість переробки, передачі та зберігання в ній інформації. Ці відомості можуть бути отримані тільки за допомогою контролю.

Контроль дає можливість своєчасно виявляти і усувати відмови, що викликають несприятливі наслідки. Можна зробити висновок, що необхідних у безвідмовності та ефективності систем найбільш доцільно домагатися за допомогою реалізації прямих методів підвищення безвідмовності апаратури спільно з її контролем і подальшим усуненням відмов і наслідків збоїв.

Організований відповідним чином контроль підвищує пристосованість системи, що перевіряється, до попередження, виявлення та усунення відмов, тобто покращує її ремонтопридатність. Отже, відновлення з малим часом контролю і ліквідації наслідків відмов і збоїв є потужним, а в ряді випадків єдиним методом підтримання необхідного рівня надійності систем при експлуатації і забезпечення необхідної достовірності переробки і передачі інформації.

Види контролю доцільно класифікувати за метою проведення; глибиною і повнотою реалізованих перевірок; за ступенем автоматизації контрольних операцій; за часом і послідовністю їх реалізації; за типом конструктивної реалізації засобів контролю та їх розташуванню щодо об'єктів перевірки; за ієрархією керування; за типом реалізованого вирішального правила; по відношенню до режимів роботи системи, що перевіряється.

Залежно від того, яка кінцева мета проведення контролю, він може бути класифікований як контроль працездатності та як діагностичний контроль. При контролі працездатності мета перевірок зводиться до

своєчасного виявлення фактів відсутності або наявності несправності в перевіреній системі і викривлень у вигляді збоїв у вихідний інформації. При діагностичному контролі перевірку проводять з метою встановлення місця і причини несправності або характеру відмови. Ці види контролю засновані на різних методах перевірки, які по-різному технічно реалізуються і використовуються в різних умовах.

У загальному випадку контроль працездатності є складовою частиною діагностичного контролю. Принципово, майже завжди можна здійснити діагностичний контроль, не маючи інформації про те, працездатна система чи ні. Однак, для реалізації діагностичного контролю потрібно більше часу, тому, як правило, спочатку виконують контроль працездатності, як більш простий і найменший, по відношенню до витрат часу. Потім, якщо це необхідно, проводять діагностичну перевірку.

За ступенем автоматизації контрольних операцій (ступеня участі оператора) контроль може бути:

– неавтоматизованим (ручним), якщо підключення вимірювальних приладів (спеціальних контрольних пристроїв) або перемикання апаратури в контрольний режим оператор проводить ручним способом. Порівняння вимірюваних параметрів з номінальними значеннями, прийняття рішення про працездатність апаратури і технічна діагностика також здійснюються людиною-оператором;

– напівавтоматизованим, якщо деяка частина операцій по підключенню і переключенню вимірювальних приладів (контрольних пристроїв) або переведення апаратури в контрольні режими реалізується автоматично, а порівняння вимірюваних параметрів з номінальними значеннями, прийняття рішення про працездатність апаратури і діагностика несправностей (відмов) і збоїв здійснюються оператором;

– автоматизованим, якщо вся послідовність контрольних операцій, включаючи вироблення сигналу про працездатність апаратури та місця виникнення (причини) несправності або збою, здійснюється без втручання

людини, тобто автоматично.

За часом реалізації контрольних операцій розрізняють періодичний контроль, який реалізується через певні інтервали часу, та оперативний (безперервний), здійснюваний безперервно, в процесі виконання системою, перед завданням, що стоїть перед нею.

За типом конструктивної реалізації контроль може бути внутрішнім і зовнішнім. Внутрішній контроль здійснюється засобами, які є складовою частиною об'єкта перевірки. При зовнішньому контролі оцінка стану об'єкта перевірки проводиться з використанням приладів, які не входять до його структури, тобто є зовнішніми по відношенню до нього. Як правило, зовнішні засоби перевірки конструктивно реалізуються у вигляді автономної системи контролю, яка може бути використана для перевірки різних об'єктів. Якщо система призначена для перевірки об'єктів одного класу, то вона називається спеціалізованою, а якщо для об'єктів кількох класів – універсальною. Прикладом перших є спеціалізовані автомати контролю, а других – системи контролю, побудовані на базі керуючих машин. Зовнішній контроль, як правило, є періодичним.

Система контролю може бути рухомою і стаціонарною (нерухомою). У першому випадку вона транспортується від одного об'єкта контролю до іншого, а в другому – стаціонарно встановлена. При цьому можливе розташування системи контролю як безпосередньо у об'єкта контролю, так і на значній відстані від нього, з використанням спеціальної лінії зв'язку. Відповідно до цього, контроль може бути безпосереднім або дистанційним (телемеханічним).

Перевірка окремих пристроїв (блоків, каналів) може здійснюватися як незалежно, так і спільно, при їх взаємодії за схемою, яка відповідає робочому режиму. У першому випадку контроль називають автономним, а в другому – комплексним. Перевірку окремих самостійних об'єктів великої системи в ряді випадків називають об'єктним контролем.

Залежно від режимів роботи системи, що перевіряється, при яких

реалізуються контрольно-перевірочні операції, розрізняють контроль в робочому режимі і профілактичний контроль. Контроль в робочому режимі здійснюється в процесі виконання системою покладених на неї функцій. Профілактичний контроль призначений для встановлення стану системи в цілому і окремих її пристроїв в період профілактичних робіт. Контроль цього виду може бути проведений при нормальному і «важкому» режимі роботи системи. Оцінка стану може проводитися як на момент контролю, так і з прогнозуванням. "Важкий" режим роботи системи дає можливість виявити елементи. які знаходяться на межі відмови, і своєчасно їх замінити. При цьому нестійкі відмови можуть стати стійкими, що полегшує їх виявлення і локалізацію.

Залежно від принципів формування і отримання ознак, за сукупністю яких оцінюється стан системи, всі згадані види контролю можуть бути реалізовані прямими і непрямими методами. Перші з них базуються на формуванні зазначених ознак за значеннями основних параметрів, що характеризують якість функціонування системи, що перевіряється, а другі – на використанні для цієї мети побічних явищ, що виникають при її функціонуванні.

До непрямих відносяться методи, що базуються на реалізації людиною-оператором евристичних вирішальних правил за сукупністю його візуальних, акустичних і тактильних сприйнятть, а також методи, засновані на комплексній оцінці результатів інструментальних вимірювань різних видів сигналів (акустичних, електромагнітних і т.д.), що відносяться до розряду побічних явищ, супутніх роботі системи, що перевіряється. До прямих відносяться програмні і апаратні методи контролю, раціональне поєднання яких дозволяє в значній мірі автоматизувати процес проведення перевірок.

При програмному контролі кількісні значення згаданих ознак визначаються завчасно і зберігаються в системі контролю або виробляються робочою схемою об'єкта контролю в період виконання ним основного завдання. При апаратному контролі вони визначаються спеціально

призначеним для цієї мети надлишковим пристроєм, який перероблює, за певним алгоритмом, робочу вхідну інформацію.

Програмний контроль заснований на реалізації спеціальних програм і логічних методів, які контролюють роботу системи в цілому або окремих її пристроїв і елементів. Залежно від способу організації програмний контроль підрозділяється на програмно-логічний, алгоритмічний і тестовий.

Програмно-логічний контроль організовується на основі використання надлишкової вихідної і проміжної інформації. Найбільш просто реалізувати програмно-логічний контроль за допомогою кількаразової переробки інформації з подальшим порівнянням отриманих результатів. Для виявлення викривлення інформації досить забезпечити дворазову її переробку. Виправлення інформації можливо тільки в тому випадку, якщо число циклів її переробки не менше трьох (мажоритарний принцип). Зазвичай при реалізації цього методу система контролю автоматично забезпечує третій цикл (і більше) переробки інформації, якщо результати перших двох не співпали.

Алгоритмічний контроль є різновидом програмно логічного контролю. При алгоритмічному контролі, на основі аналізу алгоритмів задач, що реалізуються даною системою, будується так званий «усічений» алгоритм, який використовується для цілей контролю. Усічений алгоритм повинен бути, по своїй довжині і часу виконання, приблизно на порядок менше основного алгоритму. Алгоритмічний контроль найбільш доцільно застосовувати в керуючих елементах системи, які вирішують певний клас задач, пов'язаних з керуванням реальними об'єктами. У цьому випадку розроблені основний і усічений алгоритми. Програми, що реалізують ці алгоритми, використовуються тривалий час.

Тестовий контроль – це перевірка систем за допомогою спеціальних випробувальних програм. При виконанні тесту обчислювальна машина, яка використовується в системі, здійснює певну послідовність дії над вихідними числами, порівнює отримані результати і в разі їх розбіжності, фіксує помилку.

Основне завдання при складанні тестів – найбільш повне охоплення системи, її засобів і окремих вузлів і режимів їх роботи підбором відповідних прикладів і поєднань операцій. На відміну від програмно-логічного контролю, тестовий контроль, сам по собі, не дозволяє встановлювати правильність виконання системою основної програми, так як в момент «прогону» тесту рішення покладених на систему завдань переривається. Звідси ясно, що тестовий контроль, безпосередньо в ході роботи системи, може використовуватися тільки періодично. Він грає важливу роль при ремонті системи і перевірці її функціонування.

Перевага тестового контролю, у процесі якого реалізуються контролюючі та діагностичні тести, полягає в тому, що він: дозволяє автоматизувати процес виявлення і пошуку несправностей, зводить до мінімуму час на їх усунення і не вимагає додаткової апаратури, крім деякої ємності пам'яті програм.

Застосування тільки періодичного тестового контролю знижує продуктивність системи, так як на час проходження тесту система припиняє роботу за основною програмою. Зниження продуктивності залежить від глибини охоплення тестовим контролем вузлів і пристроїв, різних засобів і елементів системи.

Апаратний контроль – це такий контроль, який функціонує безперервно у процесі всього часу роботи системи паралельно з рішенням основного об’єкту і реалізується за допомогою введеного в його структуру контрольного обладнання. За принципами практичної реалізації апаратного контролю, його можна розділити на: контроль за модулем, контроль з використанням коригувальних кодів, апаратно-мікропрограмний і мажоритарний контроль.

У зв'язку з тим, що апаратний контроль здійснюється безперервно протягом усього часу функціонування об'єкта контролю, він дозволяє виявити як несправності, так і збої в момент їх виникнення або з запізненням на одну-дві операції. Так як контрольні операції здійснюються паралельно з основним процесом обробки і передачі інформації, то апаратний контроль

практично не знижує продуктивності засобів системи.

Таким чином, до переваг апаратного контролю можна віднести його безперервність, здатність виявити як несправності (відмови), так і збої в момент їх виникнення; здатність автоматично локалізувати місце несправності з точністю до функціонального вузла (більш точна локалізація здійснюється за допомогою діагностичних тестів), а також здатність усувати наслідки збоїв безпосередньо в процесі обчислення автоматичним перекладом в режим повторення певної ділянки програми; можливості самоперевірки.

Недоліком апаратного контролю є необхідність введення додаткової контрольної апаратури, яка сама може служити джерелом несправностей і збоїв у роботі.

Комбінований контроль в різних поєднаннях може включати в себе як прямі, так і непрямі методи. Він найбільш характерний для великих систем, де чільна роль при реалізації процедури прийняття рішення відводиться людині.

Кожен з перерахованих методів і видів контролю має певні переваги і недоліки, тому в більшості випадків доцільна їх відповідна комбінація, яка визначається специфікою вирішуваних завдань і принципів побудови контрольованої системи.

Загальні принципи реалізації програмних і апаратних методів не залежать від того, є об'єкт контролю безперервним або дискретним. Однак, при практичній розробці методів перевірки функціонування сучасних великих систем, необхідно враховувати специфіку кожної з них, звертаючи увагу на особливості взаємодії, дискретних і безперервних підсистем, що входять до таких систем (пристроїв, блоків і т.д.).

Стан системи, що перевіряється, при контролі оцінюють за результатами аналізу її реакцій на різні вхідні впливу під якими розуміють зміни параметрів зовнішнього середовища і параметрів її елементів (вузлів, модулів, блоків і т.д.). Тому, при розробці методів контролю, необхідно мати

у своєму розпорядженні модель об'єкта контролю, яка б встановлювала взаємозв'язок між станами системи, що перевіряється і відповідними її реакціями.

Ця модель – аналог реального процесу – має враховувати найбільш суттєві чинники, що впливають на систему, і відображати зміну поточних значень параметрів, що характеризують її структуру і якість функціонування. Модель будують на основі ідеалізації і абстракції процесів, що протікають в об'єктах контролю. Особливості моделі, що розробляється, залежать як від постановки задачі, так і від специфіки таких процесів. У більшості випадків, первинний опис роботи об'єкта контролю прагнуть представити в формі функціональних рівнянь, логічних рівнянь або різних способів опису алгоритмів. А оскільки будь-яка велика система відрізняється значною складністю, доводиться вдаватися до різних методів наближеного її опису.

В результаті проведення контролю повинні вироблятися спеціальні керуючі впливи на контрольовані системи. Ці дії переводять систему на повторне рішення задачі (при наявності збоїв), перемикають на резерв відмовлені пристрої (при наявності несправностей), а також сигналізують технічний персонал про факт місця появи несправності. Керуючий вплив може бути реалізовано як за автоматичним командам, так і за командами людини-оператора. В обох випадках, основне призначення контролю полягає у забезпеченні необхідною інформацією (необхідною достовірністю) споживача, що приймає рішення на формування конкретних дій, що управляють. Це дозволяє забезпечити задану якість функціонування контрольованої автоматичної системи або її підсистем, блоків елементів і т.д.

1.2 Вбудовані системи діагностування в електроенергетиці

Метою діагностування є забезпечення раціональної експлуатації електрообладнання при заданих показниках надійності і скорочення витрат на технічне обслуговування і ремонт (ТОР). Ця мета досягається шляхом

управління технічним станом електрообладнання в процесі експлуатації, що дозволяє проводити ТОР відповідно до даних діагностування.

Основне завдання технічного діагностування полягає в отриманні достовірної інформації про технічний стан електрообладнання в процесі експлуатації. Вона вирішується на основі вимірювання, контролю, аналізу і обробки кількісних і якісних значень параметрів електрообладнання, а також шляхом управління обладнанням відповідно до алгоритму діагностування.

Аналіз причин виникнення дефектів електрообладнання показує, що технічний стан кожного з них характеризується як тільки йому притаманними індивідуальними, так і загальними ознаками. Для кожного виду обладнання характерні свої типові дефекти, які багаторазово зустрічаються в експлуатації. Об'єднавши всі дефекти і ознаки їх появи в окремі групи, отримаємо структуру діагностування електрообладнання, що складається з трьох рівнів і підсистем: перевірки функціонування, виявлення дефектів, оцінки і прогнозування працездатності. При цьому на кожному наступному рівні використовуються результати попередніх.

До апаратурних засобів діагностування відносяться різні пристрої: прилади, пульти, стенди, спеціальні промислові комп'ютери. Апаратурні кошти, які складають з об'єктом діагностування, конструктивно, єдине ціле, є вбудованими апаратурними засобами діагностування. Прикладами подібних засобів можуть бути електровимірювальні прилади (струму, напруги, потужності, частоти та інші), пристрої індикації технічного стану елементів (реле, світловипромінюючі діоди, неонові лампи і т.п.), пристрої контролю ізоляції та інші.

Якщо в схемах експлуатації електрообладнання непередбачено вбудовані засоби діагностування або їх виявляється недостатньо для діагностування з необхідною глибиною, то застосовують зовнішні апаратурні засоби діагностування, виконані окремо від конструкції обладнання і підключаються до нього лише в процесі діагностування. Найпростішими прикладами зовнішніх апаратурних засобів можуть бути комбіновані

прилади для вимірювання в колах постійного і змінного струму, тестери логічного стану, електронно-променеві та цифрові осцилографи, переносні вимірювальні комплекти і т.п.

Якщо апаратурні засоби діагностування призначені тільки для однотипного обладнання, то вони є спеціалізованими, а якщо для обладнання різного конструктивного виконання та функціонального призначення – універсальними.

Зовнішні спеціалізовані засоби діагностування – це пристрої, що використовуються, наприклад, для перевірки працездатності окремих елементів або вузлів електрообладнання на стадіях технічного контролю після виконання ремонтних робіт.

У число вбудованих спеціалізованих засобів діагностування можуть входити спеціально розроблені обчислювальні пристрої з жорстко запрограмованими алгоритмами діагностування конкретної системи електрообладнання.

1.3 Процедури діагностування HDL-моделей кінцевих автоматів

При існуючому різноманітті вихідних форм опису проектів цифрових пристроїв (ЦП) можна виділити найбільш популярні в світі: аналітичні - мови опису апаратури (HDL), графічні або візуальні - ієрархічні цифрові структури і схеми, граф-схеми алгоритмів операційних або керуючих пристроїв (flow chart). Одним з поширених способів вихідного опису кінцевого керуючого автомата (КА) на мові опису апаратури є автоматний шаблон, тобто спеціальна структура HDL-коду, яка будується на основі графа переходів автомата (state diagram) або прямої структурної таблиці (ПСТ). Побудова графа переходів кінцевого автомата на основі інших способів опису його функціонування є мистецтвом проектувальника і особливостями інструментальних засобів систем автоматизованого проектування радіоелектронної апаратури (САПР РЕА).

Найбільш складним і витратним етапом в сучасному циклі проектування ЦП є функціональна верифікація, тобто процес виявлення, локалізації та усунення помилок в системній моделі щодо специфікації, на що витрачається більше половини загального часу проектування. Основною формою опису проектів ЦП в САПР РЕА є мови опису апаратури, тому об'єктом верифікації є модель ЦП, написана на мові опису апаратури, тобто HDL-модель.

Можливі помилки проектування в HDL-моделях визначаються стилем опису HDL-коду. Під помилкою проектування вважається визначення помилки в HDL-операторі, яка не відноситься до класу синтаксичних і порушує алгоритм функціонування моделі пристрою, заданий специфікацією. Виділення фрагментів HDL-коду, що описують поведінку кінцевих автоматів, стилем «автоматний шаблон», дозволяє визначити помилку проектування типу «неправильний перехід у графі переходів автомата», що відповідає помилці у виборі поточного стану в операторі when, помилці вибору наступного стану в функції переходів (a_i замість a_j), помилці в операторі if() при аналізі вхідного сигналу, помилці в призначенні вихідного сигналу. Для проведення діагностичного експерименту (ДЕ) з пошуку помилок проектування реалізується стратегія обходу всіх дуг графа переходів кінцевого автомата, починаючи з початкової вершини. При цьому перевіряються всі поодинокі несправності переходів, а також справності функцій автомата, що забезпечують ці переходи [5].

ДЕ над HDL-моделлю кінцевого автомата полягає в подачі на неї вхідних впливів, відповідно до обраної стратегії обходу змістовного графа переходів, отриманні вихідних реакцій на Waveform і порівняння отриманих реакцій з еталоном. На підставі цього робиться висновок про відповідність HDL-моделі специфікації. ДЕ проводиться з використанням системи верифікації HDL-моделей (TestBench) в середовищі проектування Active-HDL. При проведенні ДЕ в простих HDL-моделях КА, подача вхідних впливів і порівняння отриманих реакцій з еталонами не представляє особливих

труднощів, навіть в режимі візуального порівняння з Waveform, так як тестові дані подаються, безпосередньо, на входи автомата, а реакції знімаються з його виходів.

Виходячи з вищесказаного, метою даної роботи є розробка автоматної моделі апаратної системи діагностування пристроїв управління в електроенергетиці і газопостачанні, представлення цієї моделі на мовах опису апаратури та їх реалізація на технологічній платформі ПЛІС.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– представити алгоритм керування пункту газорозподілу у вигляді граф-схеми алгоритму (ГСА);

– на підставі граф-схеми алгоритму, побудувати автоматну модель керуючого пристрою у вигляді графа переходів автомата та побудувати алгоритм її діагностування, шляхом обходу всіх дуг графа переходів;

– описати алгоритм діагностування моделлю на мові опису апаратури і виконати її верифікацію;

– виконати схемну реалізацію керуючого пристрою та пристрою діагностування з використанням САПР ПЛІС. Оцінити апаратурні витрати на систему діагностування.

2 МОДЕЛЬ ОБ'ЄКТА ДІАГНОСТУВАННЯ

2.1 Автоматична станція газорозподілу

Сукупність об'єкта керування та технічних пристроїв, призначених для нього, називається системою автоматичного керування (регулювання) (САК, САР). Основне завдання САК полягає в тому, щоб на основі інформації про об'єкт виробити керуючі впливи, що дозволяють підтримувати об'єкт в стабільному стані або перевести його в новий стабільний стан. Технічні пристрої, що входять до САК, включають в себе: датчики; пристрої, що визначають закон функціонування об'єкта; регулятори, що виробляють керуючі впливи по необхідному закону керування; керівні органи і виконавчі механізми.

Як об'єкт керування (ОК) розглянемо автоматичну станцію газорозподілу (АГРС), розташовану, як правило, на віддаленій території без присутності кваліфікованого персоналу. Сучасна АГРС – це комплекс обладнання, і вимірювальних приладів для регулювання розподілу газу.

Автоматична станція газорозподілу (або автоматична газорозподільна станція (АГРС)) призначена для зниження тиску газу, його одоризації, очищення від рідкої фракції і механічних домішок. АГРС працює в автоматичному режимі. З метою підвищення безпеки та надійності експлуатації обладнання на АГРС, а також для оперативного централізованого контролю технологічних параметрів, встановлено комплекс телеметрії. Комплекс телеметрії складається з керуючого автомата, набору датчиків і табло індикації.

Зовнішній вигляд одного з приміщень АГРС представлений на рис. 2.1, а функціональна схема роботи АГРС – на рис 2.2.

Рисунок 2.1 – Зовнішній вигляд одного з приміщень АГРС

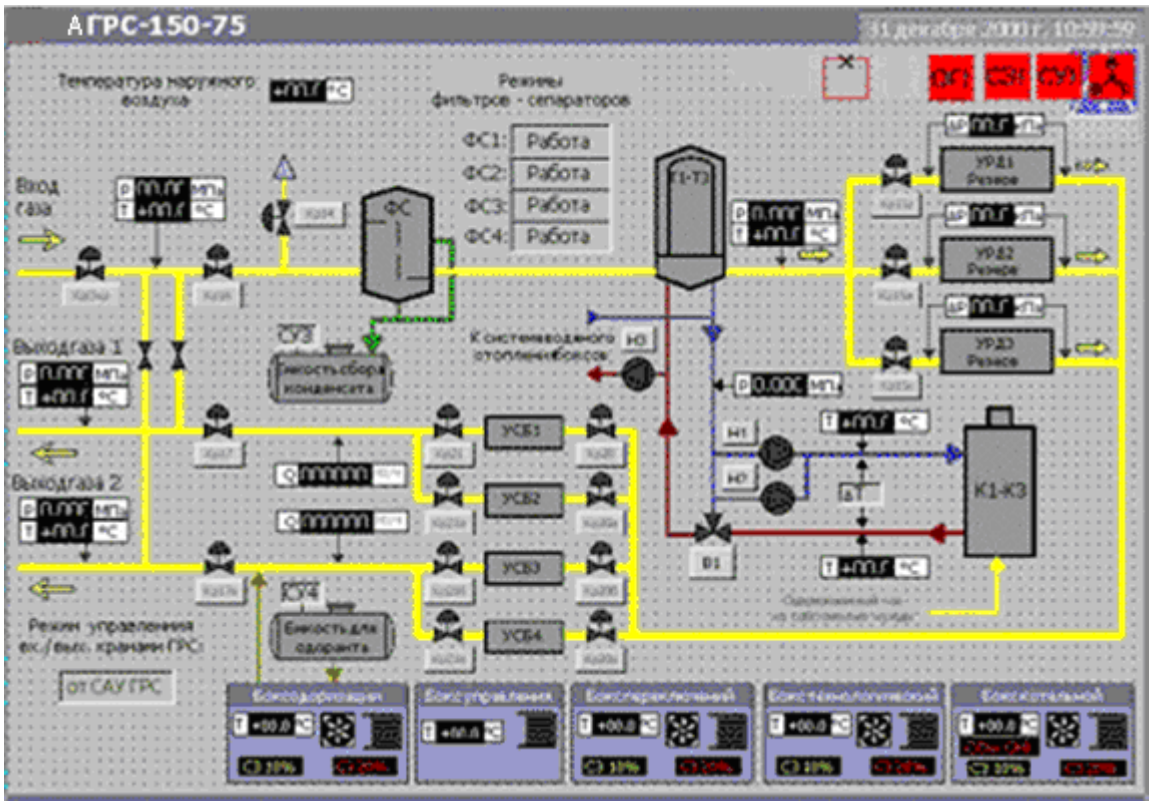


Рисунок 2.2 – Функціональна схема роботи АГРС

Інформація про роботу АГРС передається по бездротовому каналу зв'язку на цифрове табло індикації. Табло індикації складається зі

світлодіодів зеленого і червоного кольору, що відповідає робочому (зелений) і аварійному (червоний) режиму роботи обладнання АГРС.

Словесний опис спрощеного алгоритму функціонування АГРС і опис використовуваних датчиків представлено нижче:

1. Датчик вхідного тиску P_1 (X1).

Тиск на вході контролюється за допомогою аналогового датчика тиску P_1 (зняття показань Y2), який працює в діапазоні $0{,}3МПа \leq P_1 \leq 0{,}5МПа$ (X1=1). Якщо тиск більше або менше зазначеного, то X1=0 і на табло індикації буде горіти світлодіод червоного кольору (Y8).

2. Датчик температури T_1 (X2).

Датчик температури T_1 (зняття показань Y3), працює в діапазоні $10^{\circ}C \leq T_1 \leq 25^{\circ}C$ (X2=1). При збільшенні або зменшенні температури більше заданої (X2=0) на табло індикації буде переданий аварійний сигнал (Y9).

3. Датчик контролю максимального рівня газоконденсату H_1 (X3).

Невід'ємною частиною обладнання АГРС є вузол очищення газу від рідких домішок і механічних частинок. Вузол очистки складається з двох сепараторів, які підключені до системи газопроводів паралельно, що дає їм можливість працювати незалежно один від одного. У сепараторі відбувається відділення газу від рідкої фракції. Очищений газ проходить через верхню частину сепаратора в газопровід і далі надходить на регулятор тиску. Контроль максимально можливого заповнення сепаратора здійснюється за допомогою датчика H_1 (зняття показань Y4), Датчик H_1 працює в діапазоні $40\% \leq H_1 \leq 60\%$ (X3=1).

При досягненні рідкою фракцією рівня 60% – датчик сигналізує про переповнення (X3=0), і на електромагнітний соленоїдний клапан скидання рідини надходить керуючий сигнал на його відкриття (Y12). Рідина надходить в роздільну ємність для подальшої обробки та утилізації. Як тільки рівень рідини в сепараторі зменшиться до 40%, електромагнітний клапан закриється.

4. Датчик контролю мінімального рівня газоконденсату H_2 (X4)

При досягненні рівня рідини 15% спрацює датчик H_2 (зняття показань Y5) мінімального рівня (X4=0) і на табло індикації буде переданий аварійний сигнал (Y10). Зниження рівня рідини в сепараторі до 15% (X4=1) може відбутися тільки в разі зниження тиску на вході, тобто $P_1 \leq 0{,}3$МПа.

5. Датчик контролю вихідного тиску P_2 після регулятора (X5)

Регулятор тиску на АГРС забезпечує зниження тиску газу до необхідного (P_2 = 0,1МПа) і автоматично підтримує його в рамках цього тиску (зняття показань Y6). Датчик вихідного тиску P_2 після регулятора працює в діапазоні $0{,}05\text{МПа} \leq P_2 \leq 0{,}125\text{МПа}$ (X5=1). При підвищенні вихідного тиску $P_2 \leq 0{,}125$МПа (X5=0) спрацьовує запобіжний скидний клапан (Y13). Він буде відкритий до тих пір, поки не виконається умова $P_2 \leq 0{,}125$МПа. Якщо з якоїсь причини тиск не відповідає діапазону $0{,}05\text{МПа} \leq P_2 \leq 0{,}125$МПа, то на табло індикації відобразиться передача аварійного сигналу (Y14).

6. Датчик загазованості приміщення АГРС CH_4 (X6)

Загазованість приміщення АГРС може виникнути в разі витоків газу через фланцеві з'єднання або через сальникові ущільнення газового обладнання (зняття показань Y7). У безаварійному режимі X6 = 1. Вибухонебезпечна концентрація газу в приміщенні знаходиться в межах від 5% до 15%. Датчик загазованості вже при концентрації 1% (X6 = 0 при $CH_4 \geq 1\%$) передає аварійний сигнал на табло індикації (Y11).

7. Датчик приводу аварійного відкриття клапана ПСК при підвищенні тиску (X7). Датчик, контролює тільки підвищення тиску. При $P_2 \geq 0{,}125$МПа, що відповідає підвищенню тиску на 25% від робочого (X7 = 0), на табло індикації буде горіти світлодіод червоного кольору і спрацює звукова сигналізація (Y14).

2.2 Модель керуючого автомата в системі управління АГРС

На рис.2.3 представлена спрощена ГСА роботи АГРС.

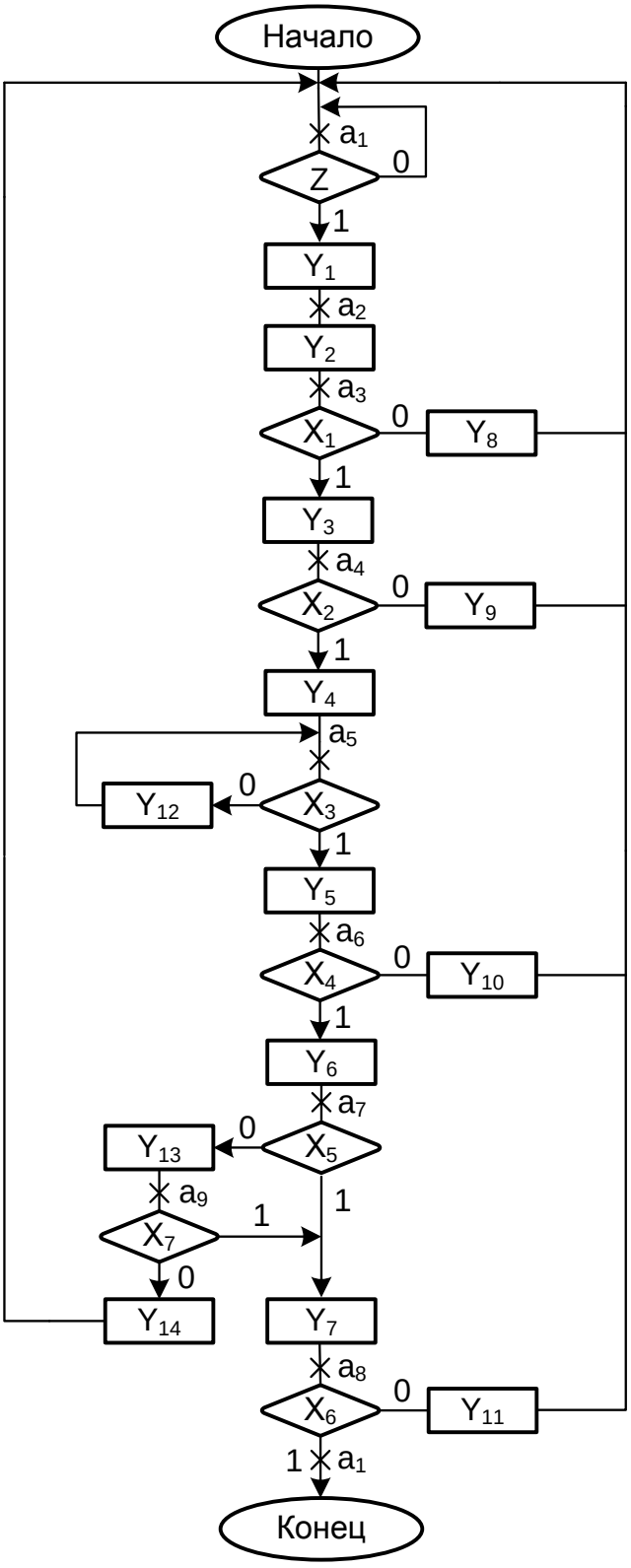


Рисунок 2.3 – Граф-схема алгоритма роботи АГРС

Для перетворення ГСА в автоматну модель, необхідно відзначити стан автомата. Як автоматну модель будемо розглядати автомат Мілі. Його

станами на ГСА позначаються входи вершин, наступних за операторними вершинами. На основі автоматної моделі Мілі будується граф переходів КА (рис. 2.4).

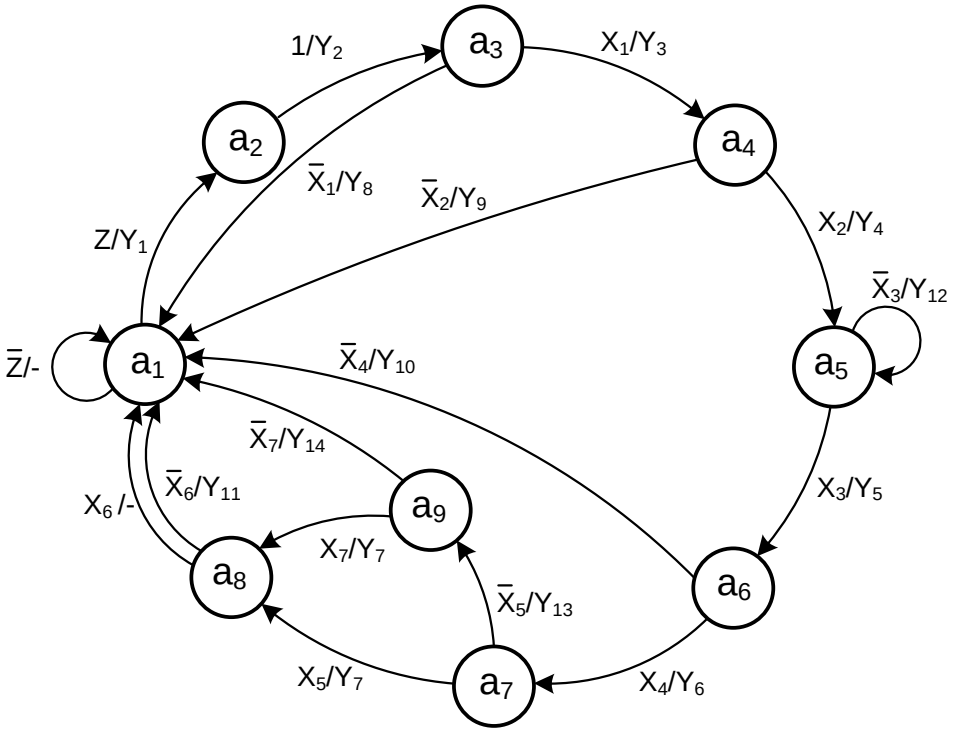


Рисунок 2.4 – Граф переходів керуючого автомата Мілі

Граф переходів автомата представляється на мові опису апаратури VHDL в формі двопроцесного автоматного шаблону. VHDL-модель автомата Мілі, представлений в лістингу 2.1.

Лістинг 2.1 – VHDL-модель керуючого автомата АГРС

```
-- Finite state machine that generate output signals for
-- control object to execute data operations

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity GAS_FSM is
     port (
```

```
          clk: in STD_LOGIC;
          reset: in STD_LOGIC;
          X: in STD_LOGIC_VECTOR (0 to 7);
          Y: out STD_LOGIC_VECTOR (1 to 14));
end GAS_FSM;

architecture GAS_FSM_arch of GAS_FSM is

     -- Not good use keyword “signal” because states always
     -- must be initialized!!!
     constant A1: std_logic_vector(3 downto 0):= "0001";
     constant A2: std_logic_vector(3 downto 0):= "0010";
     constant A3: std_logic_vector(3 downto 0):= "0011";
     constant A4: std_logic_vector(3 downto 0):= "0100";
     constant A5: std_logic_vector(3 downto 0):= "0101";
     constant A6: std_logic_vector(3 downto 0):= "0110";
     constant A7: std_logic_vector(3 downto 0):= "0111";
     constant A8: std_logic_vector(3 downto 0):= "1000";
     constant A9: std_logic_vector(3 downto 0):= "1001";
     signal State, NextState: std_logic_vector (3 downto 0);

begin
     State_CurrentState: process (clk,reset)
     begin
          if rising_edge(clk) then
               if reset='1' then   State <= A1;
               else State <= NextState;
               end if;
          end if;
     end process;

     State_NextState: process (X, State)
     begin
          Y<= (others=>'0');
          case State is
          -- X(0) is a bit that is marked on algorithm graph as Z
               when "0001" =>
                    if (X(0)='1') then
                         NextState <= A2;
                         Y(1) <= '1';
                    elsif (X(0) = '1') then
                         NextState <= A1;
                    else NextState <= A1;
               end if;

               when "0010" =>
                         NextState <= A3;
                         Y(2) <= '1';

               when "0011" =>
                    if (X(1)='1') then
                         NextState <= A4;
                         Y(3) <= '1';
```

```
     elsif (X(1) = '0') then
          NextState <= A1;
          Y(8) <= '1';
     else NextState <= A1;
     end if;

when "0100" =>
     if (X(2)='1') then
          NextState <= A5;
          Y(4) <= '1';
     elsif (X(2) = '0') then
          NextState <= A1;
          Y(9) <= '1';
     else NextState <= A1;
     end if;

when "0101" =>
     if (X(3)='1') then
          NextState <= A6;
          Y(5) <= '1';
     elsif (X(3)='0') then
          NextState <= A5;
          Y(12) <= '1';
     else NextState <= A1;
     end if;

when "0110" =>
     if (X(4)='1') then
          NextState <= A7;
          Y(6) <= '1';
     elsif (X(4) = '0') then
          NextState <= A1;
          Y(10) <= '1';
     else NextState <= A1;
     end if;

when "0111" =>
     if (X(5)='1') then
          NextState <= A8;
          Y(7) <= '1';
     elsif (X(5)='0') then
          NextState <= A9;
          Y(13) <= '1';
     else NextState <= A1;
     end if;

when "1000" =>
     if (X(6)='0') then
          NextState <= A1;
          Y(11) <= '1';
     elsif (X(6) = '1') then
          NextState <= A1;
     else NextState <= A1;
```

```
                    end if;

               when "1001" =>
                    if (X(7)='1') then
                         NextState <= A8;
                         Y(7) <= '1';
                    elsif (X(7) = '1') then
                         NextState <= A1;
                         Y(14) <= '1';
                    else NextState <= A1;
                    end if;
               -- No null for synthesis!!!
               when others => NextState <= A1;

          end case;

     end process;

end GAS_FSM_arch;
```

На рис.2.5 показаний режим роботи КА, який описує послідовність переходів A1-A2-A3-A4-A5-A6-A1. Перехід зі стану в стан відповідає за ініціацію блоків і датчиків, описаних в лістингу 2.1. Відбувається ініціація автомата (запуск різних блоків для початку роботи УА (A1 - A2; X(0) = 1 => Y1 = 1)) опитування датчика тиску (A3-A4; Y2 = 1), температури (A4-A5; X(1) = 1 => Y3 = 1) та газоконденсату (A5-A6; X(3) = 1 => Y4 = 1). Після опитування кожного датчика відбувається порівняння їх показників з необхідними для подальшої роботи. Позитивна перевірка (X(1) = X(2) = X(3) = 1) показує, що показники не перевищують допустимі і автомат далі продовжує коректну роботу. Перевірка наступного датчика мінімального рівня газоконденсату не проходить (A6-A7; X(4) = 0) і виникає аварійна ситуація (Y(10) = 1), яка повертає автомат в початковий стан (A1).

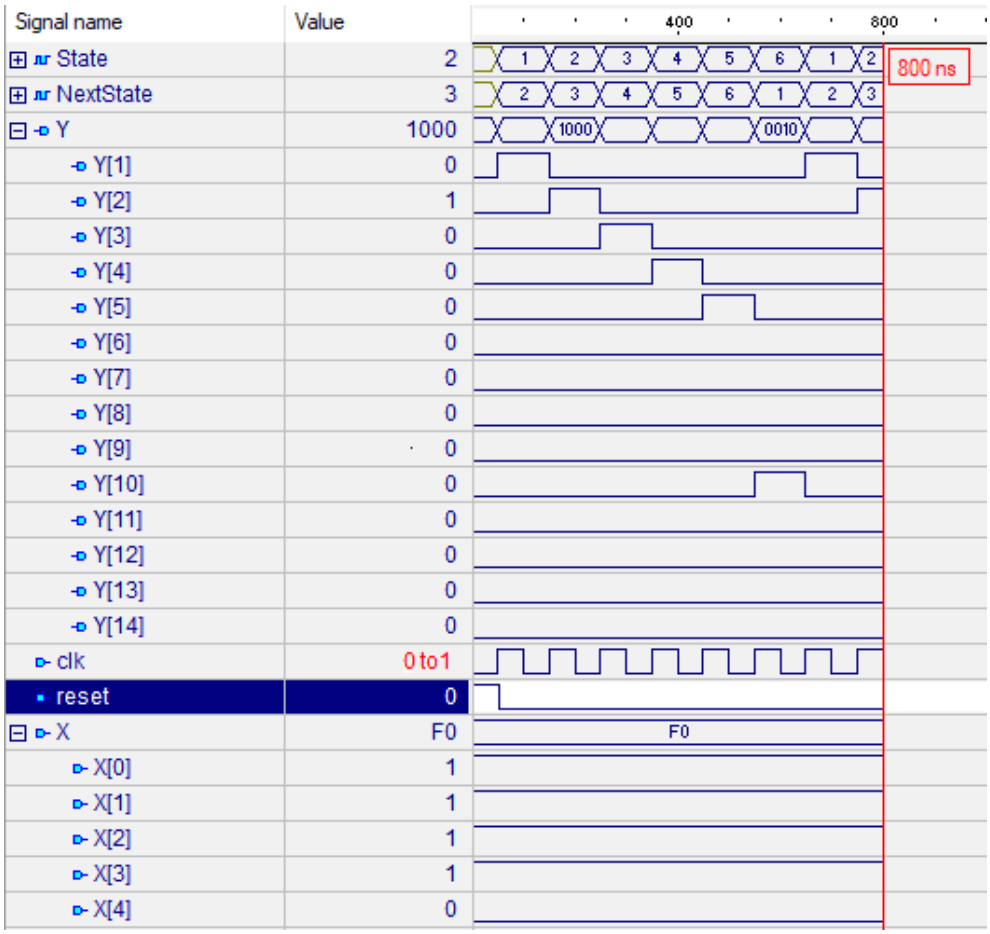


Рисунок 2.5 – Waveform роботи КА АГРС

Апаратурні витрати по синтезу моделі керуючого пристрою засобами САПР XILINX ISE в ПЛІС Spartan 3E наведені в таблиці 2.1.

Таблиця 2.1 – Результати синтезу керуючого автомата

Selected Device : 3s500efg320-5	Варіант 1
Number of Slices	10 out of 4656
Number of Slice Flip Flops	9 out of 9312
Number of 4 input LUTs:	18 out of 9312
Number of IOs	24
Number of bonded IOBs:	24 out of 232
Number of GCLKs	1

3 МОДЕЛЬ ВБУДОВАНОГО ПРИСТРОЮ ДІАГНОСТУВАННЯ

3.1 Стратегія діагностування керуючого автомата

ДЕ над HDL-моделлю кінцевого автомата полягає в подачі на неї вхідних впливів, відповідно до обраної стратегії обходу змістовного графа переходів, отриманні вихідних реакцій на Waveform і порівняння отриманих реакцій з еталоном. На підставі цього робиться висновок про відповідність HDL-моделі специфікації. ДЕ проводиться з використанням системи верифікації HDL-моделей (TestBench) в середовищі проектування Active-HDL. При проведенні ДЕ в простих HDL-моделях КА, подача вхідних впливів і порівняння отриманих реакцій з еталонами не представляє особливих труднощів, навіть в режимі візуального порівняння з Waveform, так як тестові дані подаються, безпосередньо, на входи автомата, а реакції знімаються з його виходів.

На підставі стратегії обходу всіх дуг графа КА (рис. 2.4) будується алгоритм діагностування з гарантованою повнотою щодо одиночних несправностей переходів (наприклад, перехід a1 – a2 замість a1 – a3), представлений на рис.3.1 у вигляді бінарного дерева рішень [5].

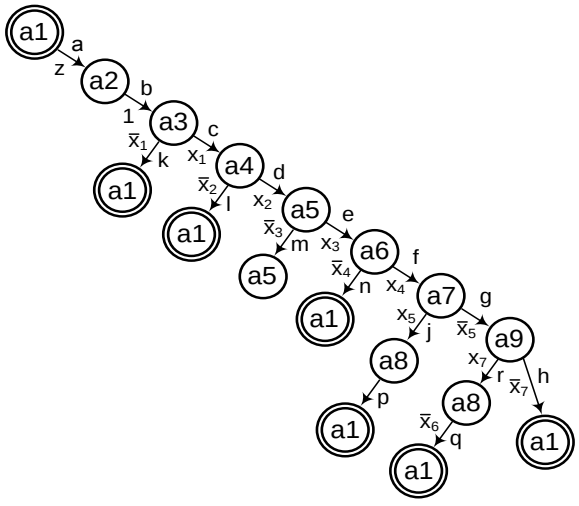


Рисунок 3.1 – Дерево рішень для графа переходів КА

По дереву рішень будуються варіанти обходу дуг графа. При цьому слід враховувати, що проводиться так званий «неруйнівний ДЕ», коли обхід дуг графа починається з початкової вершини і в ній же закінчується. Варіанти обходу графа переходів КА представлені на рис.3.2.

a1 – a2 – a3 – a1;
a1 – a2 – a3 – a4 – a1;
a1 – a2 – a3 – a4 – a5 – a6 – a1;
a1 – a2 – a3 – a4 – a5 – a6 – a7 – a8 – a1;
a1 – a2 – a3 – a4 – a5 – a6 – a7 – a9 – a1;
a1 – a2 – a3 – a4 – a5 – a6 – a7 – a9 – a8 – a1.

Рисунок 3.2 – Варіанти обходу графа переходів КА

Варіанти обходу графа переходів КА заносяться в енергонезалежну пам'ять і використовуються при проведенні ДЕ з використанням апаратного пристрою діагностування.

3.2 VHDL-моделі пристрою діагностування

3.2.1 Структура та алгоритм пристрою діагностування

При діагностуванні керуючого пристрою (КП) АГРС, що працює з механічними, електричними, термодинамічними датчиками, виникає проблема реалізації вхідних значень електричної напруги, тиску, температури, які виникають при реальній роботі АГРС. Виходячи з цього, пропонується імітувати вихідні значення відповідних датчиків в двійковому алфавіті {0, 1} (1 – параметр знаходиться в заданому допустимому інтервалі, 0 – параметр вийшов за межі допустимих значень).

Таким чином, пропонується апаратна реалізація пристрою

діагностування (ПД), яке забезпечує виконання всіх переходів по графу керуючого автомата (КА), тобто фактично реалізує його пряму структурну таблицю [7]. Структурна схема об'єкта керування з пристроєм діагностування представлена на рис. 3.3, де: УУ – КП – керуючий пристрій; ОУ – ОК – об’єкт керування; УД – ПД – пристрій діагностування; РгД – регістр даних, куди заноситься черговий варіант обходу графа; У – Y – керуючі сигнали; ХХ – сповіщальні сигнали об'єкта керування (в нашому випадку логічно оброблені показники датчиків); Х – сповіщальні сигнали для керуючого пристрою; УХ – YX – «імітація» сповіщувальних сигналів пристроєм діагностування; ТМ (test mode) – режим роботи приладу (ТМ = 0 – робота в режимі керування, ТМ = 1 – робота в режимі діагностування).

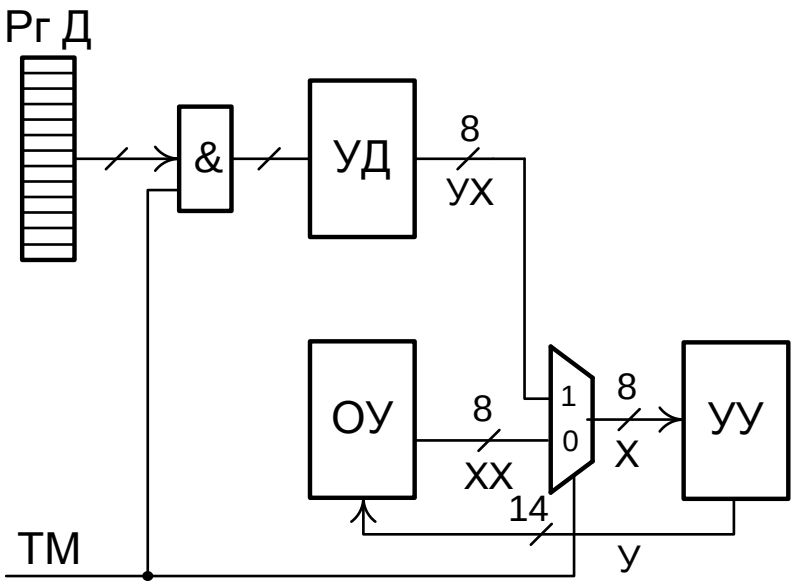


Рисунок 3.3 – Апаратний пристрій діагностування

Суть проведення діагностічного експерименту з використанням ПД полягає в наступному. При підготовці ДЕ будуються шляхи обходу графа переходів КА. При ТМ = 1 на вхід ПД надходить послідовність станів, автомата (рис. 3.2), переходи в які необхідно перевірити (порядок опитування відповідних датчиків, який позначається при подачі в КП масивом Х), і на виході ПД буде формуватися послідовність сповіщувальних сигналів YX, які

«імітують» роботу об'єкта керування (ОК). Сигнали YX через мультиплексор надходять на КП (в форматі сповіщувальних сигналів X, що надходять з об'єкта управління) та ініціюють його роботу. Керуючий пристрій, в свою чергу, формує вихідні сигнали Y, які надходять на ОК.

Вихідні сигнали Y або порівнюються з еталонами (якщо такі є) в режимі відключених датчиків, або, відповідним чином, відображаються на панелі індикації. Якщо послідовність Y збігається з еталоном або індикація відображається вірно, то КП працює коректно.

Розглянемо реалізацію алгоритму діагностування. У ньому реалізується модифікована пряма структурна таблиця КА, яка представлена на рис. 3.4.

state	nextstate	X	YX(0-7)
a1	a2	Z=1	10000000
a2	a3	1	00000000
a3	a4	X1=1	01000000
	a1	X1=0	00000000
a4	a5	X2=1	00100000
	a1	X2=0	00000000
a5	a6	X3=1	00010000
	a5	X3=0	00000000
a6	a7	X4=1	00001000
	a1	X4=0	00000000
a7	a8	X5=1	00000100
	a9	X5=0	00000000
a8	a1	X6=0	00000000
	a1	X6=1	00000010
a9	a8	X7=1	00000001
	a1	X7=0	00000000

Рисунок 3.4 – Реалізація алгоритму діагностування

Схемна реалізація КП може здійснюватися кількома способами, в залежності від способу кодування вхідної послідовності станів автомата при обході графа переходів. Розглянемо ці варіанти.

3.2.2 Бітове кодування алгоритму діагностування

За варіантом 1, послідовність станів автомата, відповідна черговому варіанту обходу графа переходів КП кодується бітовим масивом, кожна ланка якого відповідає стану. Якщо біт стану дорівнює 1, то перехід відбувається до наступного стану, позначеного 1. Перехід між станами описаний у вигляді звичайного двопроцесного шаблону. Наприклад, якщо D(2) = 1 (D(2) відповідає стану A2), то наступним станом і буде A2. Нуль в ланці масиву відповідає поверненню в початковий стан. Схематичне зображення послідовності a1 – a2 – a3 – a4 – a5 – a6 – a1 представлено на рис.3.5. Дане зображення містить повний шлях обходу ГСА КА. Відсутність бітів «-» означає, що ця ланка не несе ніякої ролі, тому що відбулося повернення в початковий стан (A7 = 0).

A1	A2	A3	A4	A5	A6	A7	A8	A9	A1
1	1	1	1	1	1	0	-	-	-

Рисунок 3.5 – Схематичне зображення вхідного масиву КП за варіантом 1

Лістингу 3.1 представдяє HDL-модель пристрою діагностування з кодуванням вхідного масиву за варіантом 1.

Лістинг 3.1 – VHDL-модель КП за варіантом 1

```
--The diagnostic device that generates a set of conditions for
the operation of the control device
--a set of conditions (8-bit register X) is a check of the
sequence of transitions
--on which the control automaton must pass to check for
```

```
correctness    working

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity GAS_DD_FSM is
     port (
          clk: in STD_LOGIC;
          reset: in STD_LOGIC;
          D: in STD_LOGIC_VECTOR (1 to 10);
          Yx: out STD_LOGIC_VECTOR (0 to 7));
end GAS_DD_FSM;

architecture arch of GAS_DD_FSM is

     signal State, NextState: std_logic_vector(3 downto 0);
     -- Not good use keyword “signal” because states always
     -- must be initialized!!!
     constant A1: std_logic_vector(3 downto 0):= "0001";
     constant A2: std_logic_vector(3 downto 0):= "0010";
     constant A3: std_logic_vector(3 downto 0):= "0011";
     constant A4: std_logic_vector(3 downto 0):= "0100";
     constant A5: std_logic_vector(3 downto 0):= "0101";
     constant A6: std_logic_vector(3 downto 0):= "0110";
     constant A7: std_logic_vector(3 downto 0):= "0111";
     constant A8: std_logic_vector(3 downto 0):= "1000";
     constant A9: std_logic_vector(3 downto 0):= "1001";
     -- Signal for copying input sequence from external memory
     signal DD: std_logic_vector (1 to 10);

begin
     State_CurrentState: process (clk,reset)
     begin
          if rising_edge(clk) then
               if reset='1' then
               -- Copying input sequence from external memory
                    DD <= D;
                    State <= A1;
               else State <= NextState;
               end if;
          end if;
     end process;

     -- Sensitivity list must include control signals
     -- (State for case enumeration and input register
     -- for transit condition
     State_NextState: process (State,DD)
     begin
          Yx<= (others=>'0');
          case State is

               when "0001" =>
```

```
                if(DD(2) = '1') then
                     NextState <= A2;
                     Yx(0) <= '1';
                elsif (DD(2) = '0') then
                     NextState <= A1;
                else   NextState <= A1;
                end if;

           when "0010" =>
                     NextState <= A3;

           when "0011" =>
                if(DD(4) = '1') then
                     NextState <= A4;
                     Yx(1) <= '1';
                elsif(DD(4) = '0') then
                     NextState <= A1;
                else NextState <= A1;
                end if;

           when "0100" =>
                if(DD(5) = '1') then
                     NextState <= A5;
                     Yx(2) <= '1';
                elsif (DD(5) = '0') then
                     NextState <= A1;
                else NextState <= A1;
                end if;

           when "0101" =>
                if(DD(6) = '1') then
                     NextState <= A6;
                     Yx(3) <= '1';
                elsif (DD(6) = '0') then
                     NextState <= A5;
                else NextState <= A1;
                end if;

           when "0110" =>
                if(DD(7) = '1') then
                     NextState <= A7;
                     Yx(4) <= '1';
                elsif (DD(7) = '0') then
                     NextState <= A1;
                else NextState <= A1;
                end if;

      -- WARNING! we can transit from A7 only to A8 or A9
           when "0111" =>
                if(DD(8) = '1') then
                     NextState <= A8;
                     Yx(5) <= '1';
                elsif (DD(8) = '0') then
```

```
                    NextState <= A9;
                else NextState <= A1;
                end if;

    --WARNING! problem with a multiple transition from A8 to A1
            --when "1000" =>
            --if(D(9) = "0001") then

            when "1001" =>
                if(DD(9) = '1') then
                    NextState <= A8;
                    Yx(7) <= '1';
                elsif(DD(9) = '0') then
                    NextState <= A1;
                else NextState <= A1;
                end if;
            -- No null for synthesis machine template!!!
            when others => NextState <= A1;
        end case;

    end process;

end arch;
```

В лістингу 3.2 представлений TestBench введення вхідної послідовності станів a1 – a2 – a3 – a4 – a5 – a6 – a1 за першим варіантом кодування, а на рис. 3.6 – діаграма часу (Waveform) моделювання даної послідовності в середовищі Active-HDL.

Лістинг 3.2 – Фрагмент TestBench для введення вхідної послідовності за варіантом 1

```
        -- Unit Under Test port map
           UUT : gas_dd_fsm
             port map (
                  clk => clk,
                  reset => reset,
                  D => D,
                  Yx => Yx
             );

           clk_process: process
           begin
             clk <= '0';
             wait for 50 ns;

             clk<='1';
```

```
            wait for 50 ns;
            wait for 0 ns;

          end process;

          some_process:process
          begin
            reset<='1';

            -- Each bit of input array characterizes not crash
transition ('1') between state and crash transition ('0')

            D<="1111110000";

            wait for 60 ns;
            reset <= '0';
               wait;
          end process;…
```

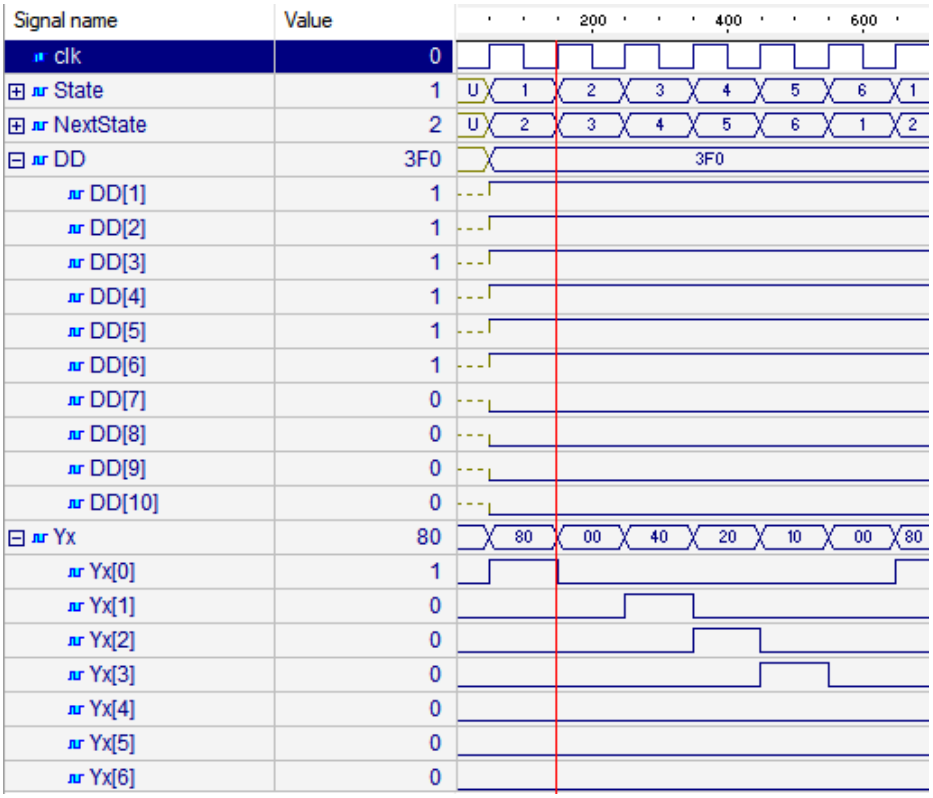


Рисунок 3.6 – Waveform роботи КП з бітовим масивом, з масивом, що описує послідовність (A1-A2-A3-A4-A5-A6-A1)

3.2.3 Алгоритм діагностування за кодами станів автомата

За варіантом 2, опис КП, що приймає на вхід масив 4-х розрядних

чисел, кожне з яких є описом стану (наприклад «0001» відповідає стану A1 і т.д.). Як тільки виявлено вектор подальшого стану, рівний початковому («0001»), то послідовність вважається завершеною і відбувається повернення в початковий стан. Якщо виявлена послідовність не є кінцевою ланкою масиву, то наступні ігноруються. Перехід між станами аналогічний першому варіанту. Схематичне зображення послідовності a1 – a2 – a3 – a4 – a5 – a6 – a1 представлено на рис.3.7. Воно відображає співвідношення стану до значення в ланці вхідного масиву.

A1	A2	A3	A4	A5	A6	A7	A8	A9	A1
0001	0010	0011	0100	0101	0110	0001	-	-	-

Рисунок 3.7 – Схематичне зображення вхідного масиву і роботи КП за варіантом 2

Лістинг 3.3 – VHDL-модель КП за варіантом 2

```
--The diagnostic device that generates a set of conditions for
the operation of the control device
--a set of conditions (8-bit register X) is a check of the
sequence of transitions
--on which the control automaton must pass to check for
correctness    working
-- The feature of device is concrete state that can not be
changed
-- (this code can optimized using ROM)

library IEEE;
use IEEE.std_logic_1164.all;
--Package for creating array of states
--Each state is signal of std_logic_vector
--(state is an item of array)
PACKAGE heap_arr_pkg IS
     type TestSequence is array (1 to 10) of std_logic_vector (1
to 4);
END;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

--Use package for creating array of states (array of
```

```
std_logic_vector)
USE work.heap_arr_pkg.all;

entity GAS_DD_Listing3 is
     port (
          clk: in STD_LOGIC;
          reset: in STD_LOGIC;
          D: in TestSequence;
          Yx: out STD_LOGIC_VECTOR (0 to 7));
end GAS_DD_Listing3;

architecture arch of GAS_DD_Listing3 is

     signal State, NextState: std_logic_vector(3 downto 0);
     -- Not good use keyword “signal” because states always
     -- must be initialized!!!
     constant A1: std_logic_vector(3 downto 0):= "0001";
     constant A2: std_logic_vector(3 downto 0):= "0010";
     constant A3: std_logic_vector(3 downto 0):= "0011";
     constant A4: std_logic_vector(3 downto 0):= "0100";
     constant A5: std_logic_vector(3 downto 0):= "0101";
     constant A6: std_logic_vector(3 downto 0):= "0110";
     constant A7: std_logic_vector(3 downto 0):= "0111";
     constant A8: std_logic_vector(3 downto 0):= "1000";
     constant A9: std_logic_vector(3 downto 0):= "1001";
     -- Signal for copying input sequence from external memory
     signal DD: TestSequence;

begin

     State_CurrentState: process (clk,reset)
     begin
          if rising_edge(clk) then
               if reset='1' then
               -- Copying input sequence from external memory
                    DD <= D;
                    State <= A1;
               else State <= NextState;
               end if;
          end if;
     end process;

     -- Sensitivity list must include control signals
     -- (State for case enumeration and input register
     -- for transit condition)
     State_NextState: process (State, DD)
     begin
          Yx<= (others=>'0');
          case State is

               when "0001" =>
                    if(DD(2) = "0010") then
                         NextState <= A2;
```

```
                    Yx(0) <= '1';
               else NextState <= A1;
               end if;

          when "0010" =>
               NextState <= A3;

          when "0011" =>
               if(DD(4) = "0100") then
                    NextState <= A4;
                    Yx(1) <= '1';
               else NextState <= A1;
               end if;

          when "0100" =>
               if(DD(5) = "0101") then
                    NextState <= A5;
                    Yx(2) <= '1';
               else NextState <= A1;
               end if;

          when "0101" =>
               if(DD(6) = "0110") then
                    NextState <= A6;
                    Yx(3) <= '1';
               elsif (DD(6) = "0101") then
                    NextState <= A5;
               else NextState <= A1;
               end if;

          when "0110" =>
               if(DD(7) = "0111") then
                    NextState <= A7;
                    Yx(4) <= '1';
               else NextState <= A1;
               end if;

     -- WARNING! we can transit from A7 only to A8 or A9
          when "0111" =>
               if(DD(8) = "1000") then
                    NextState <= A8;
                    Yx(5) <= '1';
               elsif (DD(8) = "1001") then
                    NextState <= A9;
               else NextState <= A1;
               end if;

--WARNING! problem with a multiple transition from A8 to A1
          --when "1000" =>
          --if(D(9) = "0001") then

          when "1001" =>
               if(DD(9) = "1000") then
```

```
                    NextState <= A8;
                    Yx(7) <= '1';
               else NextState <= A1;
               end if;

          -- No null for synthesis machine template!!!
          when others => NextState <= A1;
     end case;

end process;

end arch;
```

Лістинг 3.4 представляє TestBench введення вхідної послідовності станів a1 – a2 – a3 – a4 – a5 – a6 – a1 за другим варіантом кодування, а на рис. 3.8 - діаграма часу (Waveform) моделювання даної послідовності в середовищі Active-HDL.

Лістинг 3.4 – Фрагмент TestBench для введення вхідної послідовності за варіантом 2

```
-- Unit Under Test port map
     UUT : gas_dd_fsm_listing_3
          port map (
               clk => clk,
               reset => reset,
               D => D,
               Yx => Yx
          );
     clk_process: process
     begin
          clk <= '0';
          wait for 50 ns;
          clk<='1';
          wait for 50 ns;
          wait for 0 ns;
     end process;

     some_process:process
     begin
          reset<='1';
-- Each item of input array is 4-bits field
-- Each 4-bits field means current state of test sequence
          D(1)<= "0001";
          D(2)<= "0010";
          D(3)<= "0011";
```

```
        D(4)<= "0100";
        D(5)<= "0101";
        D(6)<= "0110";
        -- It means the end of transitions
        -- It returnes to initial state
        D(7)<= "0001";
        -- No matter what is in these items, because
        -- current transition endings at D(7) = "0001"
        D(8)<= "0010";
        D(9)<= "0011";
        D(10)<= "0001";

        wait for 60 ns;

        reset <= '0';
           wait;
end process;…
```

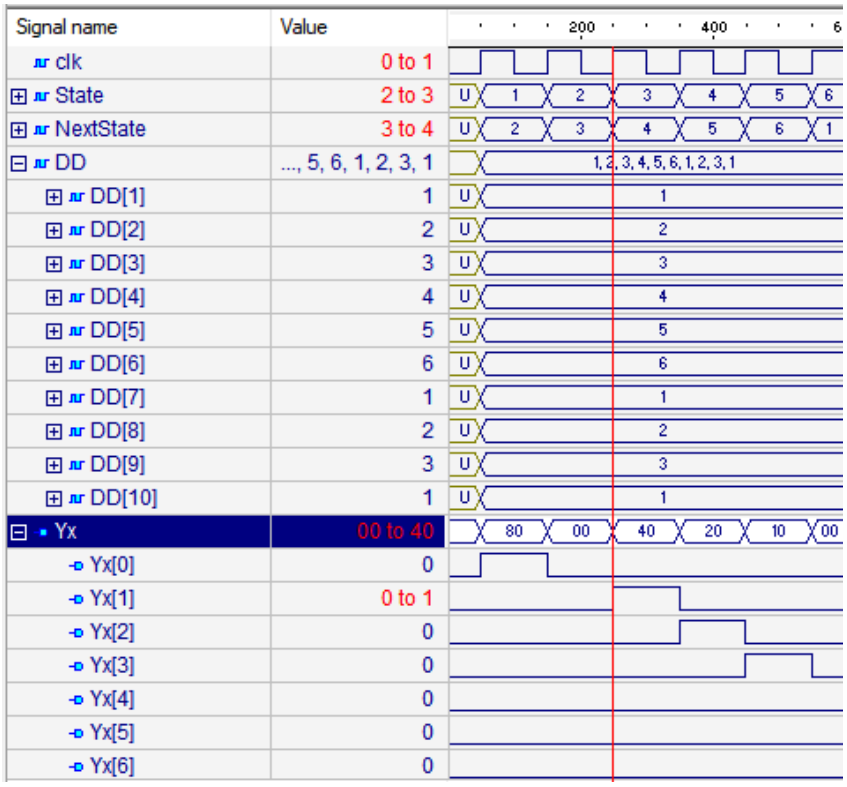


Рисунок 3.7 – Waveform ПД з використанням масива векторів, що містять стан, описаний числом (A1-A2-A3-A4-A5-A6-A1)

3.2.4 Алгоритм діагностування з використанням регістру зсуву

Варіантом 3 є КП, що приймає на вхід масив 4-х розрядних чисел, кожне з яких є описом стану. Перший процес відповідає за копіювання

вхідного масиву в проміжний і присвоєння початкових значень сигналам поточного (вихідного – A1) і подальшого станів з вхідного масиву (для того, щоб виключити подвійне присвоєння одних і тих же значень). Після присвоєння сигналам початкових значень, з переднім фронтом синхросигналу відбувається зсув вліво на один елемент масиву та присвоєння одиниці для уникнення переходів, які не відповідають послідовності. Якщо наступний стан рівний вихідному (початковому – A1), то послідовність вважається завершеною. Схематичне зображення на рис.3.8.

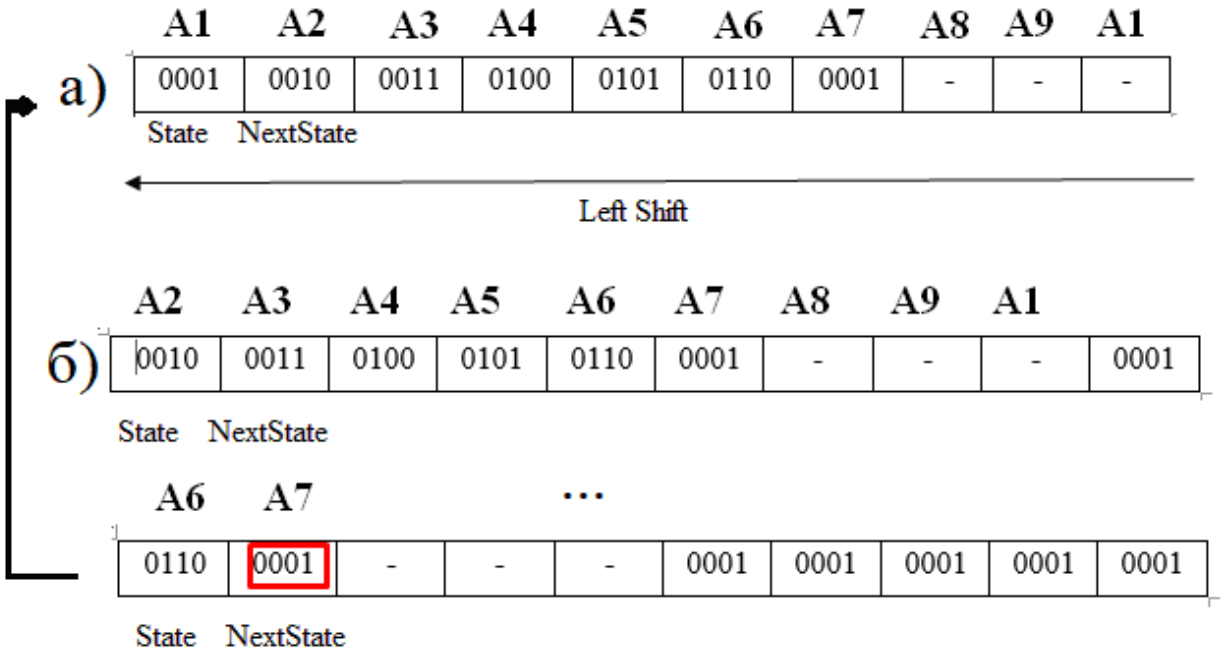


Рисунок 3.8 – Схематичне зображення вхідного масива та роботи КП з ним за варіантом 3

Лістинг 3.5 – VHDL-модель КП за варіантом 3

```
--The diagnostic device that generates a set of conditions for
the operation of the control device
--a set of conditions (8-bit register X) is a check of the
sequence of transitions
--on which the control automaton must pass to check for
correctness    working

library IEEE;
use IEEE.std_logic_1164.all;
```

```
--Package for creating array of states
--Each state is signal of std_logic_vector
PACKAGE heap_arr_pkg IS
     type TestSequence is array (1 to 10) of std_logic_vector (1
to 4);
     -- Describe sequence that ROM keeps
     -- For example it's only one sequence
     constant c_Test_sequence: TestSequence :=
("0001","0010","0011","0100","0101","0110","0001","0001","0001",
"0001");
END;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

--Use package for creating array of states (array of
std_logic_vector)
USE work.heap_arr_pkg.all;

entity GAS_Diagnostic_Device is
     port (
          clk: in STD_LOGIC;
          reset: in STD_LOGIC;
          -- No input for reading from ROM
          --D: in TestSequence;
          Yx: out STD_LOGIC_VECTOR (0 to 7));
end GAS_Diagnostic_Device;

architecture arch of GAS_Diagnostic_Device is

     -- If waveform has 'U' in starting point
     --(when reset is 1 and before rising edge)
     -- State and NextState can be initialized there
     signal State: std_logic_vector(1 to 4);
     signal NextState: std_logic_vector(1 to 4);
     --Signal for copying test sequence (reading constant)
     signal DD : TestSequence;

begin
     --Process for initialization and switching DD array
     State_CurrentState: process (clk,reset,DD)
     begin
          if rising_edge(clk) then
               --ATTENTION! Reset (reset) must load
               --only first sequence in ROM every time
               if (reset='1') then

               --Copying the first sequence from ROM to signal
                    DD <= c_Test_sequence;

     -- Synchronize process must not working with signals
```

```
-- that operated in other process (State, NextState)
-- so NextState has a role DD(2)
-- Array item needs to be checked if it initial state or not
          elsif(DD(2) = "0001") then

--Copying from ROM to signal
--ATTENTION!If we using all sequences there must be shifting
--an array of all sequences (another sequence except first)
               DD <= c_Test_sequence;

          else
               DD <= DD(2 to 10)&"0001";
          end if;

          --Initiation State and NextState after shifting
          State <= DD(1); NextState <= DD(2);
     end if;
end process;

State_NextState: process (State, NextState)
begin
     Yx<= (others=>'0');
     case State is

          when "0001" =>
               if(NextState = "0010") then
                    Yx(0) <= '1';
               end if;

          when "0011" =>
               if(NextState = "0100") then
                    Yx(1) <= '1';
               end if;

          when "0100" =>
               if(NextState = "0101") then
                    Yx(2) <= '1';
               end if;

          when "0101" =>
               if(NextState = "0110") then
                    Yx(3) <= '1';
               end if;

          when "0110" =>
               if(NextState = "0111") then
                    Yx(4) <= '1';
               end if;

          when "0111" =>
               if(NextState = "1000") then
                    Yx(5) <= '1';
               end if;
```

```
     --WARNING! problem with a multiple transition from A8 to A1
                    --when "1000" =>
                    --   if(NextState = "0001") then

               when "1001" =>
                    if(NextState = "1000") then
                         Yx(7) <= '1';
                    end if;

               -- In this case null is synthesized
               -- BUT, desirable, use another operator
               -- or operation for avoiding non-synthesize
               when others => null;
          end case;
     end process;
end arch;
```

Лістинг 3.6 представляє TestBench введення вхідної послідовності станів a1 – a2 – a3 – a4 – a5 – a6 – a1 за третім варіантом кодування, а на рис. 3.9 – Waveform моделювання даної послідовності в середовищі Active-HDL.

Лістинг 3.6 – Фрагмент TestBench для введення вхідної послідовності за варіантом 3

```
…
-- Unit Under Test port map
     UUT : gas_diagnostic_device
          port map (
               clk => clk,
               reset => reset,
               D => D,
               Yx => Yx
          );

     clk_process: process
     begin

          clk <= '0';
          wait for 50 ns;
          clk<='1';
          wait for 50 ns;
          wait for 0 ns;
     end process;

     some_process:process

     begin
          reset<='1';
```

```
-- Each item of input array is 4-bits field
-- Each 4-bits field means state of test sequence
-- First element of array is State and the second is NextState

        D(1)<= "0001";
        D(2)<= "0010";
        D(3)<= "0011";
        D(4)<= "0100";
        D(5)<= "0101";
        D(6)<= "0110";

        -- It means the end of transitions
        -- It returnes to initial state

        D(7)<= "0001";

        -- No matter what is in these items, because
        -- current transition endings at D(7) = "0001"
        D(8)<= "0010";
        D(9)<= "0011";
        D(10)<= "0001";

        wait for 60 ns;
        reset <= '0';
           wait;

    end process;
```

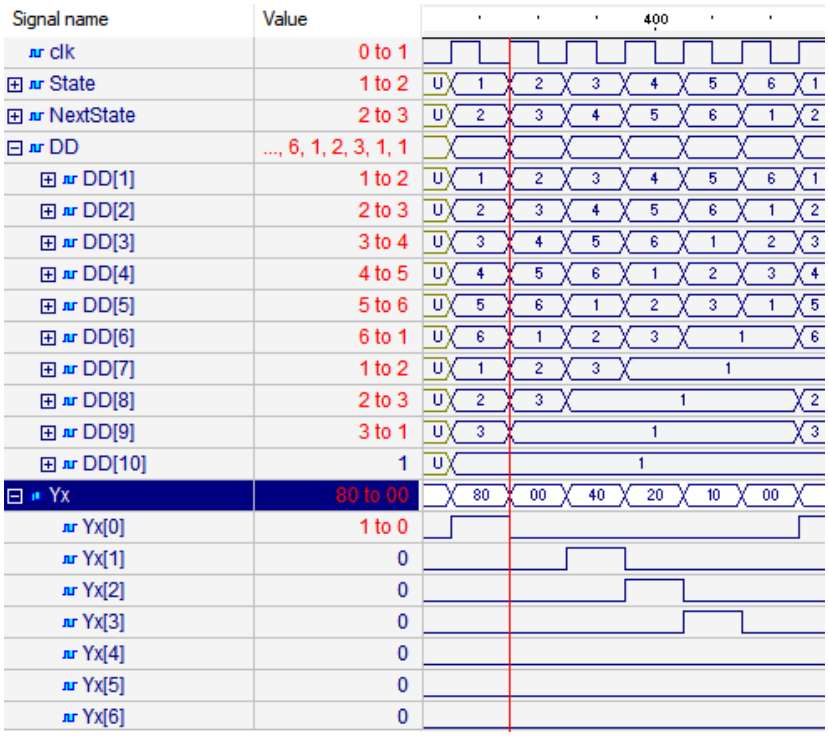


Рисунок 3.9 – Waveform ПД з використанням зсуву вхідної послідовності (для A1-A2-A3-A4-A5-A6-A1)

3.2.5 Аналіз витрат апаратури

Апаратурні витрати по синтезу різних варіантів моделей ПД засобами САПР XILINX ISE в ПЛІС Spartan 3E наведені в таблиці 3.1.

Таблиця 3.1 – Результати синтезу ПД (Device utilization summary)

Selected Device : 3s500efg320-5	Варіант 1	Варіант 2	Варіант 3
Number of Slices	13 out of 4656	27 out of 4656	23 out of 4656
Number of Slice Flip Flops	16 out of 9312	16 out of 9312	35 out of 9312
Number of 4 input LUTs:	15 out of 9312	21 out of 9312	17 out of 9312
Number of IOs	20	50	10
Number of bonded IOBs:	17 out of 232	39 out of 232	10
IOB Flip Flops:	7	28	10
Number of GCLKs	1	1	1

Аналіз результатів синтезу різних варіантів схемних реалізацій ПД показав наступне:

1. Апаратні витрати на реалізацію ПД за першим варіантом мінімальні, але при такому способі кодування послідовності станів автомата виникають проблеми, якщо граф переходів є мультиграфом, тобто між парою вершин є більше однієї дуги, переходи за якими визначаються різними сповіщувальними сигналами (для розглянутого автомата це пара станів a8 – a1).

2. В схемній реалізації ПД за варіантом 3 є приблизно в 2 рази більше тригерів (Slice Flip Flops), ніж в інших варіантах реалізації. Це обумовлено тим, що в цьому варіанті вхідний масив `c_Test_sequence` копіюється в регістр `DD`, що фактично імітує роботу з зовнішньою енергонезалежною пам'яттю.

3. З точки зору масштабованості ПД щодо числа станів автомата, третій

варіант схемної реалізації є кращим, тому що зі збільшенням довжини послідовності для обходу графа автомата апаратні витрати в цьому варіанті ростуть в незначній кількості, чого не можна сказати про варіант 2.

4. Апаратні витрати на ПД при першій-ліпшій нагоді схемної реалізації можна порівняти з апаратними витратами на КП (табл.2.1), що підтверджує правильність обраної методики побудови апаратного пристрою діагностування.

3.3 Загальна модель керуючого пристрою і діагностування

З'єднання КП і ПД являє собою певну систему управління і діагностування пристрою або, навіть, окремої системи. КП може працювати окремо від ПД, займаючись контролем роботи системи або пристрою, у вигляді подачі вихідних сигналів, що ініціюють виконання операцій. Використання повної системи управління і діагностування є тестовим режимом перевірки поведінки пристрою або системи.

У лістингу 3.7 представлений HDL-код, що описує дану систему. Це структурна модель, що складається з двох компонентів. Компонент КП пов'язаний з компонентом ПД регістром X. Це вихідний масив ПД і вхідний масив КП для генерації значень перевірки умов.

Лістинг 3.7 – Опис структурної моделі КП та ПД

```
-- Control and diagnostic system
-- Contains Diagnostic and Control devices
-- Shows system working process and can execute test mode

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Workstation is

     port(
               clk,reset : in STD_LOGIC;
               Y:out STD_LOGIC_VECTOR(1 to 14)
          );
end Workstation;
```

```
architecture Workstation of Workstation is

     component GAS_FSM is
          port(
                    clk: in STD_LOGIC;
                    reset: in STD_LOGIC;
                    X: in STD_LOGIC_VECTOR (0 to 7);
                    Y: out STD_LOGIC_VECTOR (1 to 14)
               );
     end component GAS_FSM;

     component GAS_Diagnostic_Device is
          port(
                    clk: in STD_LOGIC;
                    reset: in STD_LOGIC;
                    Yx: out STD_LOGIC_VECTOR (0 to 7)
                );
     end component GAS_Diagnostic_Device;

     signal Yx:STD_LOGIC_VECTOR(0 to 7);
begin

     UUT0:GAS_Diagnostic_Device
          port map(clk=>clk,reset=>reset,Yx=>Yx);

     UUT2:GAS_FSM
          port map(clk=>clk,reset=>reset,X=>Yx,Y=>Y);
end Workstation;
```

Побудовані VHDL-моделі керуючого і діагностичного автоматів повинні бути верифіковані інструментальними засобами системи моделювання Active-HDL. Результати верифікації аналізуються по діаграмі часу (waveform), на якій візуально відображається закон функціонування зазначених автоматів. На рис. 3.10 представлений результат моделювання роботи ПД по реалізації варіанту обходу графа a1 – a2 – a3 – a4 – a5 – a6 – a1. Результат порівняння роботи КП і ПД представлений на рис.3.11, а на рис.3.12 представлена повна Waveform з реалізацією алгоритму діагностування, що підтверджує їх повну ідентичність.

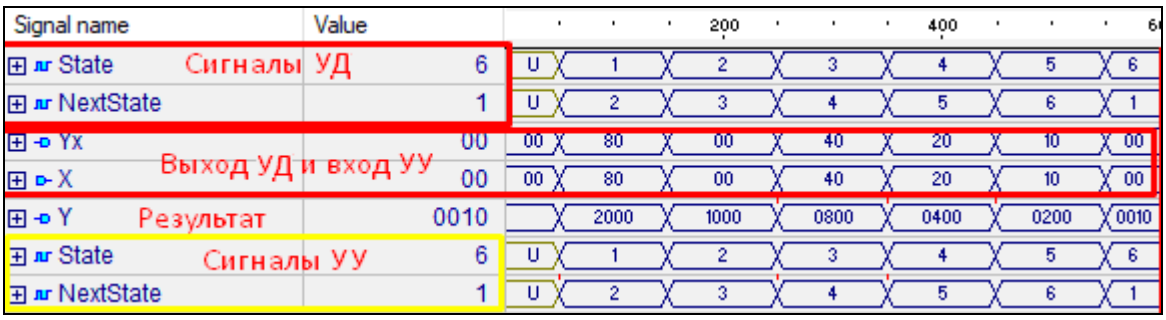


Рисунок 3.10 – Детальний опис сигналів структурної моделі

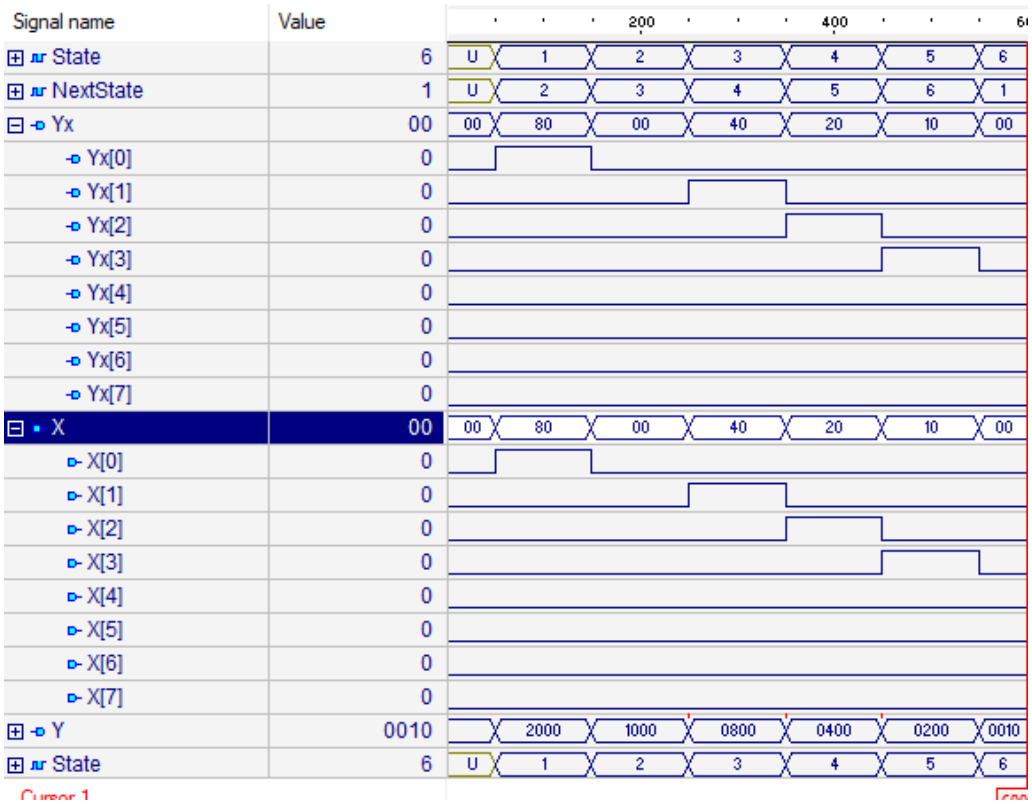


Рисунок 3.11 – Порівняння виходу ПД та входу КП

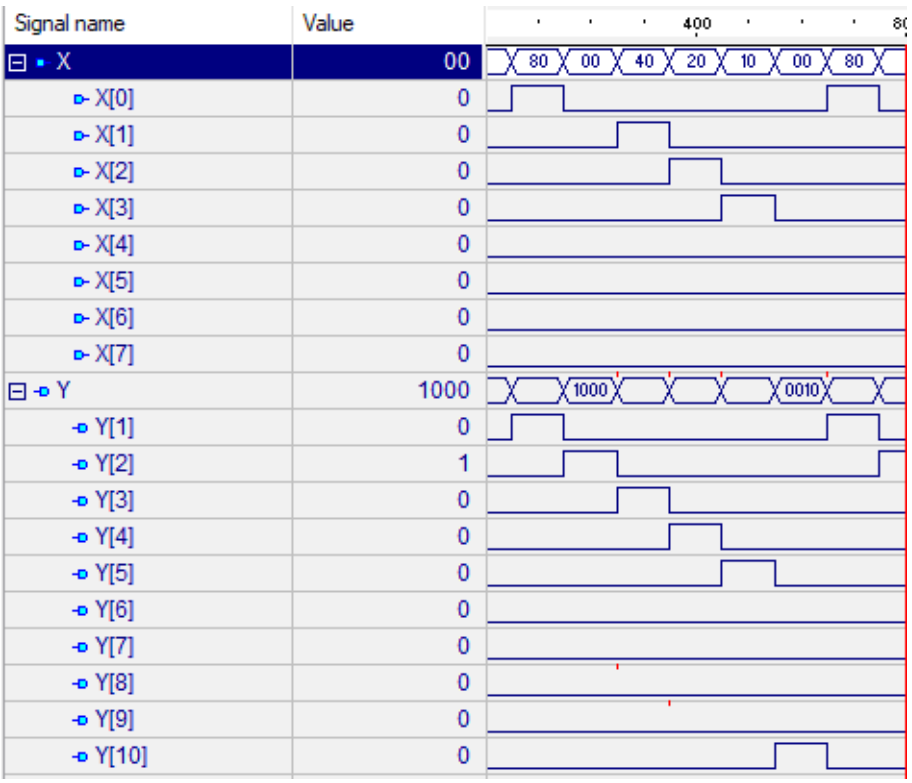


Рисунок 3.12 – Результат роботи пристроїв керування та діагностування (послідовність A1-A2-A3-A4-A5-A6-A1)

Загальні апаратні витрати по синтезу моделі КП и ПД засобами САПР XILINX ISE в ПЛІС Spartan 3E наведені в таблиці 3.2

Таблиця 3.2 – Результати синтезу по моделі КП и ПД

Selected Device : 3s500efg320-5	Варіант 1
Number of Slices	34 out of 4656
Number of Slice Flip Flops	44 out of 9312
Number of 4 input LUTs:	38 out of 9312
Number of IOs	16
Number of bonded IOBs:	17 out of 232
Number used as Shift registers: 1	1
Number of GCLKs	1

ВИСНОВКИ

У роботі представлений метод автоматизованого комп'ютерного проектування цифрового пристрою локального управління (регулювання) в електроенергетиці і газопостачанні. Як приклад обраний контур регулювання автоматичної газорозподільної станції. На основі спрощеного алгоритму функціонування АГРС, представленого ГСА, складається автоматна модель у вигляді графа переходів кінцевого автомата Мілі. Граф переходів представляється на мові опису апаратури VHDL в формі двопроцесного автоматного шаблону. По графу переходів будується алгоритм діагностування пристрою управління за стратегією обходу всіх дуг графа, що гарантує його повноту. На підставі алгоритму діагностування будується VHDL-модель апаратного пристрою діагностування. Верифікація моделі розробленого пристрою діагностування виконується з використанням системи моделювання Active-HDL. Синтез пристрою управління і пристрою діагностування виконується за допомогою САПР XILINX ISE. Результати синтезу показали, що апаратні витрати на пристрої управління і діагностування можна порівняти.

Подальші напрями досліджень можуть бути пов'язані з автоматизацією побудови алгоритму діагностування інструментальними засобами Active-HDL і створення повного циклу автоматизованого проектування пристроїв управління і діагностування.

Дана робота демонструє можливості застосування сучасних САПР ПЛІС при проектуванні цифрових систем управління і діагностування в електроенергетиці і газопостачанні.

Аналіз схемної реалізації (синтезу) HDL-моделей пристрою керування (керуючого автомата – КА) і апаратних засобів діагностування (діагностичного автомата – пристрій діагностування – ПД) дозволяє оцінити величину додаткових витрат апаратури за допомогою критерію Квайна.

Аналіз додаткових витрат апаратури показав, що їх можна порівняти з апаратурними витратами на сам пристрій управління. З одного боку, це збільшує апаратурні витрати на цифрову систему керування в цілому, але, з іншого боку, при реалізації системи керування на ПЛІС, на кристалі, як правило, присутня незадіяна програмована логіка, яка може використовуватися під апаратуру діагностування.

Якщо казати про зниження додаткових витрат апаратури на пристрій діагностування, то при автоматизованому проектуванні цифрових систем управління слід застосовувати методи тестопридатного проектування та вбудованого діагностування [8].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тоценко В.Г. Алгоритмы технического диагностирования цифровых устройств / В.Г. Тоценко. – М.: Радио и связь, 1985. – 240 с.

2. Stroud C. E. A designer’s guide to built-in self-test / Charles E. Stroud. – Kluwer Academic Publishing, 2002 – 319 p.

3. Мирошник М.А. Проектирование диагностической инфраструктуры вычислительных систем и устройств на ПЛІС: монография / М.А. Мирошник. – Х.: ХУПС, 2012. – 188 с.

4. Дианов В.Н. Диагностика и надежность автоматических систем: учебное пособие.– М.: Изд-во МГИУ, 2005. – 160 с.

5. Шкиль А.С. Диагностирование HDL-моделей микропрограммных автоматов / А.С. Шкиль, Э.Н. Кулак, А.С. Серокурова // АСУ и приборы автоматики.– 2015.– Вып. 172.– С. 22-31.

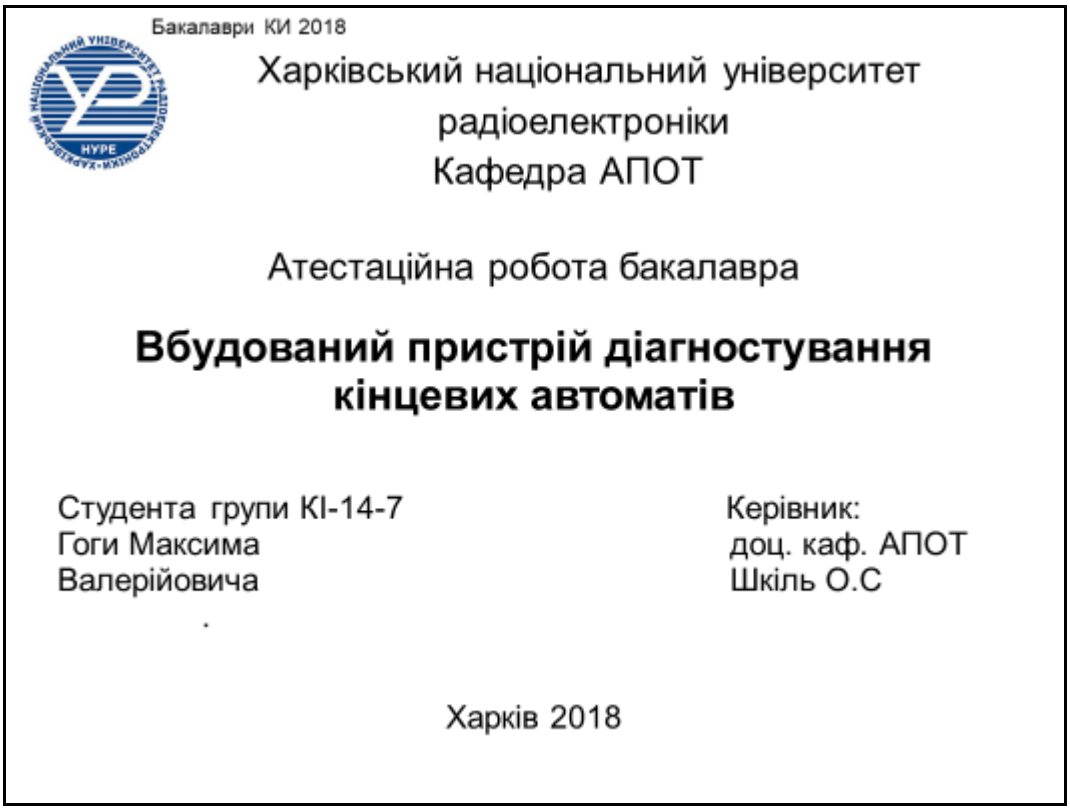
6. Нубарян С.М. Автоматизация систем теплогазоснабжения и вентиляции: Краткий курс лекций / С.М. Нубарян. - Харьков: ХНАГХ, 2007 – 147 с.

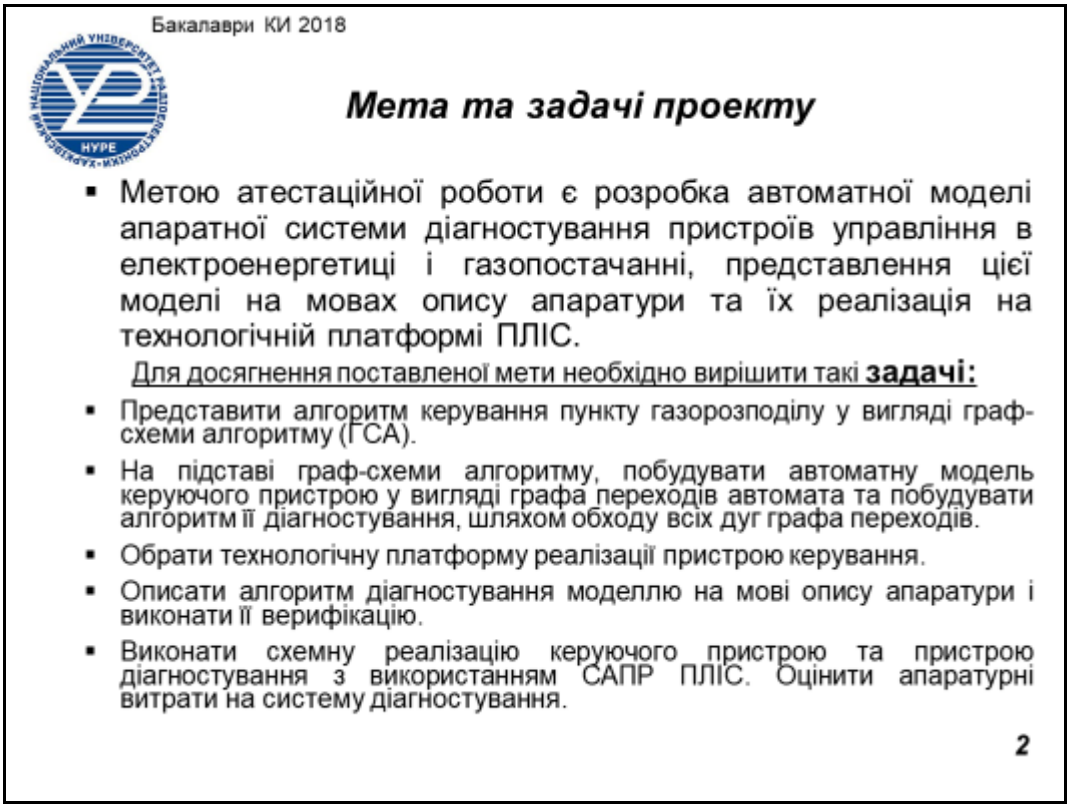
7. Баранов С.И. Синтез микропрограммных автоматов (граф-схемы и автоматы). – 2-е изд., перераб. и доп. / С.И. Баранов – Л.: Энергия, 1979. – 232 с.

8. Shkil A. Design Automation of Testable Finite State Machines / M.Miroschnyk, Y. Pakhomov, E. German, A., E. Kulak, D. Kucherenko // Proceedings of the International Sympos. EWDTS'2017, September 29-October 2, 2017 – Novi Sad, Serbia, 2017. – P.203–208.

ДОДАТОК А

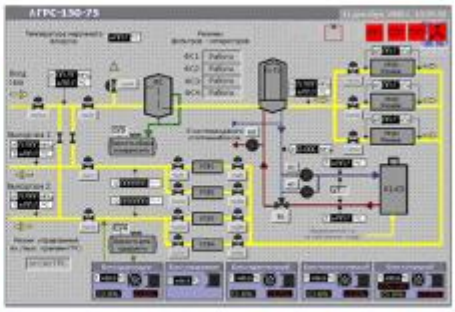
Графічна частина проекту



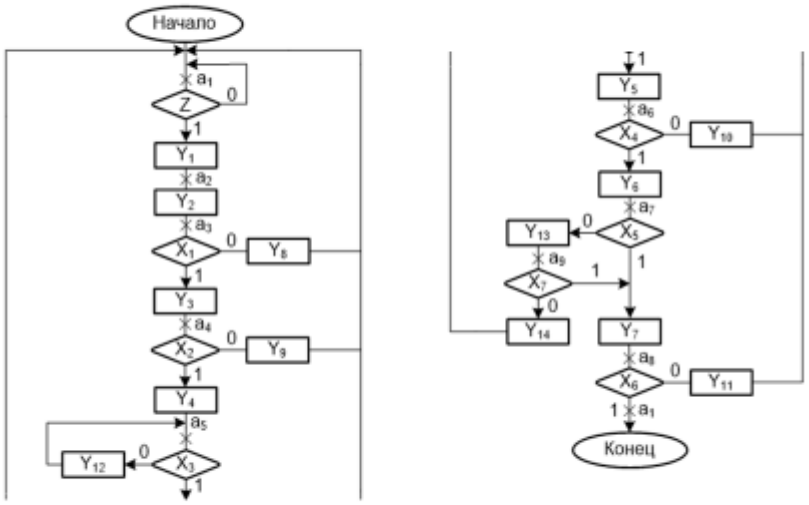


Об'єкт управління на віддаленій території

1. В електроенергетиці і газопостачанні досить поширені віддалені пункти управління, які працюють без / або з мінімальною участю людини, а також без використання персональних комп'ютерів.
2. Пристрої управління реалізуються на технологічній базі МК, ASIC або ПЛІС.
3. Процес діагностування пристрою управління має йти в автономному режимі при відключенні систем управління на досить короткий час.



Модель керуючого автомата в системі управління АГРС



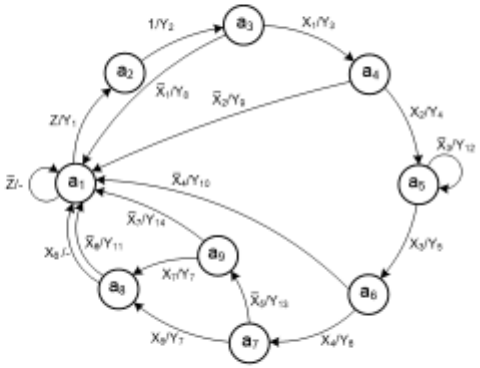
Граф-схема алгоритма роботи АГРС

Модель керуючого автомата в системі управління АГРС

Фрагмент VHDL-моделі керуючого автомата АГРС

```
State_CurrentState: process (clk,reset)
        begin
                if rising_edge(clk) then
                        if reset='1' then   State <= A1;
                        else State <= NextState;
                        end if;
                end if;
        end process;

        State_NextState: process (X, State)
        begin
                Y<= (others=>'0');
                case State is
                -- X(0) is a bit that is marked on algorithm
graph as Z
                        when "0001"=>
                                if (X(0)='1') then
                                        NextState <= A2;
                                        Y(1) <= '1';
                                elsif (X(0) = '1') then
                                        NextState <= A1;
                                else NextState <= A1;
                                end if;
```

Граф переходів керуючого автомата Мілі



Бакалаври КИ 2018

Модель керуючого автомата в системі управління АГРС

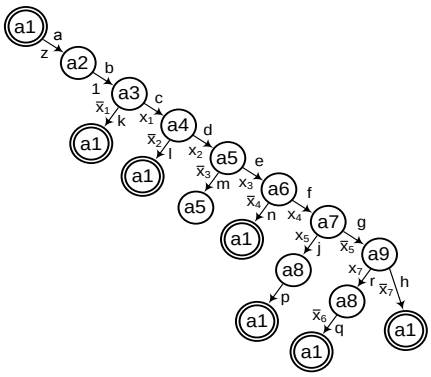
Waveform роботи КА АГРС

Selected Device : 3s500efg320-5	Варіант 1
Number of Slices	10 out of 4656
Number of Slice Flip Flops	9 out of 9312
Number of 4 input LUTs:	18 out of 9312
Number of IOs	24
Number of bonded IOBs:	24 out of 232
Number of GCLKs	1

Результати синтезу керуючого автомата

- Режим роботи, показаний на Waveform, демонструє роботу керуючого пристрою, на прикладі послідовності A1-A2-A3-A4-A5-A6-A1.
- Перехід зі стану в стан відповідає за ініціацію блоків і датчиків.

Діагностичний експеримент обходу графа



Дерево рішень для графа переходів КА

- На підставі стратегії обходу всіх дуг графа КА будується алгоритм діагностування з гарантованою повнотою щодо одиночних несправностей переходів (наприклад, перехід a1 – a2 замість a1 – a3), бінарного дерева рішень.

Варіанти обходу графа переходів КА

a1 – a2 – a3 – a1;

a1 – a2 – a3 – a4 – a1;

a1 – a2 – a3 – a4 – a5 – a6 – a1;

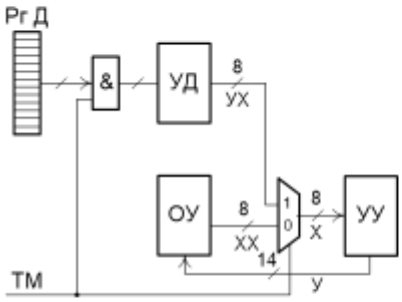
a1 – a2 – a3 – a4 – a5 – a6 – a7 – a8 – a1;

a1 – a2 – a3 – a4 – a5 – a6 – a7 – a9 – a1;

a1 – a2 – a3 – a4 – a5 – a6 – a7 – a9 – a8 – a1.

- По дереву рішень будуються варіанти обходу дуг графа. При цьому слід враховувати, що проводиться так званий «неруйнівний ДЕ», коли обхід дуг графа починається з початкової вершини і в ній же закінчується.

Структура пристрою діагностування



Апаратний пристрій діагностування

Структурна схема об'єкта керування з пристроєм діагностування представлена на рисунку зліва, де:

- УУ – КП – керуючий пристрій;
- ОУ – ОК – об'єкт керування;
- УД – ПД – пристрій діагностування;
- РгД – регістр даних, куди заноситься черговий варіант обходу графа;
- У – Y – керуючі сигнали;
- XX – сповіщальні сигнали об'єкта керування (в нашому випадку логічно оброблені показники датчиків);
- X – сповіщальні сигнали для керуючого пристрою;
- УХ – YX – «імітація» сповіщувальних сигналів пристроєм діагностування;
- ТМ (test mode) – режим роботи приладу (ТМ=0 – робота в режимі керування, ТМ= 1 – робота в режимі діагностування).

Реалізація алгоритму діагностування

state	Nextstate	X	YX(0-7)
a1	a2	Z=1	10000000
a2	a3	1	00000000
a3	a4	X1=1	01000000
	a1	X1=0	00000000
a4	a5	X2=1	00100000
	a1	X2=0	00000000
a5	a6	X3=1	00010000
	a5	X3=0	00000000
a6	a7	X4=1	00001000
	a1	X4=0	00000000
a7	a8	X5=1	00000100
	a9	X5=0	00000000
a8	a1	X6=0	00000000
	a1	X6=1	00000010
a9	a8	X7=1	00000001
	a1	X7=0	00000000

- Алгоритм діагностування. реалізується за допомогою модифікованої прямої структурної таблиці керуючого пристрою (КП).
- Схемна реалізація КП може здійснюватися кількома способами, в залежності від способу кодування вхідної послідовності станів автомата при обході графа переходів.

Обхід графу переходів (бітове кодування)

A1	A2	A3	A4	A5	A6	A7	A8	A9	A1
1	1	1	1	1	1	0	-	-	-

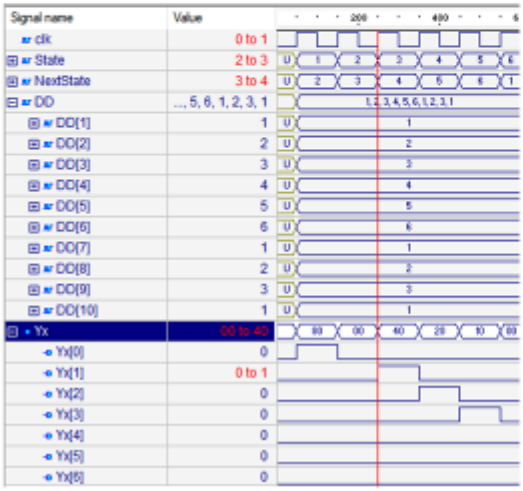
Схематичне зображення вхідного масиву КП

- Приклад вхідної послідовності станів a1 – a2 – a3 – a4 – a5 – a6 – a1.
- Дешевий спосіб реалізації алгоритму діагностування з точки зору апаратури.
- При такому способі кодування послідовності станів автомата виникають проблеми, якщо граф переходів є мультиграфом, тобто між парою вершин є більше однієї дуги, переходи за якими визначаються різними сповіщувальними сигналами.

Обхід графу переходів (за кодами станів автомата)

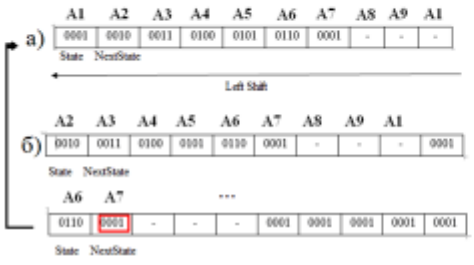
A1	A2	A3	A4	A5	A6	A7	A8	A9	A1
0001	0010	0011	0100	0101	0110	0001	-	-	-

Схематичне зображення вхідного масиву КП

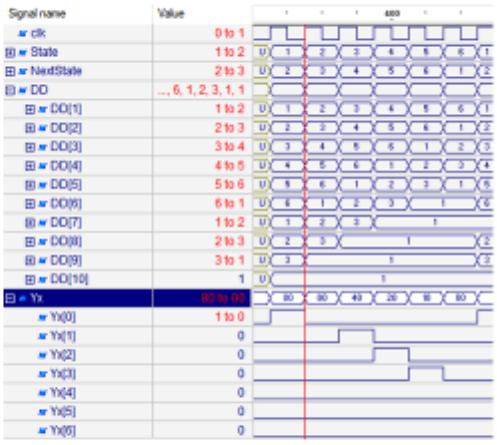


- Приклад вхідної послідовності станів a1 – a2 – a3 – a4 – a5 – a6 – a1.
- Спрощене сприйняття роботи алгоритму, так як кожне 4-х розрядне число є описом стану.
- Перехід між станами аналогічний варіанту з бітовим кодуванням.
- Самий витратний спосіб реалізації алгоритму діагностування

Обхід графу переходів (з використанням регістру зсуву)



Схематичне зображення вхідного масива та роботи КП з ним,



Waveform ПД з використанням зсуву вхідної послідовності

- Вхідний масив копіюється в регістр даних, що фактично представляє роботу з зовнішньою енергонезалежною пам'яттю.
- Усуває недолік попереднього варіанту реалізації алгоритму діагностування в плані масштабованості пристрою діагностування щодо числа станів автомата.

VHDL-модель обходу графа переходів (з використанням регістру зсуву)

Фрагмент VHDL-моделі КП

```
--Process for initialization and switching DD array
      State_CurrentState: process (clk,reset,DD)
      begin
            if rising_edge(clk) then
                  --ATTENTION!  Reset (reset) must
load
                  --only first sequence in ROM every
time
                  if (reset='1') then

                  --Copying the first sequence from
ROM to signal
                        DD <= c_Test_sequence;

      -- Synchronize process must not working with
signals
      -- that operated in other process (State,
NextState)
      -- so NextState has a role DD(2)
      -- Array item needs to be checked if it initial
state or not
                  elsif(DD(2) = "0001") then
```

```
--Copying from ROM to signal
      --ATTENTION!If  we using all sequences
there must be shifting
      --an array of all sequences (another
sequence except first)
                  DD <= c_Test_sequence;

            else DD <= DD(2 to 10)&"0001";

            end if;

            --Initiation State and NextState
after shifting
            State <= DD(1); NextState <=
DD(2);
      end if;
end process;
```


Порівняння результатів синтезу

Selected Device : 3s500efg320-5	Варіант 1
Number of Slices	10 out of 4656
Number of Slice Flip Flops	9 out of 9312
Number of 4 input LUTs:	18 out of 9312
Number of IOs	24
Number of bonded IOBs:	24 out of 232
Number of GCLKs	1

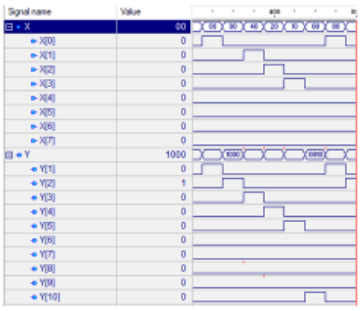
Порівняння результатів синтезу керуючого автомата та автомата діагностування

Selected Device : 3s500efg320-5	Варіант 1	Варіант 2	Варіант 3
Number of Slices	13 out of 4656	27 out of 4656	23 out of 4656
Number of Slice Flip Flops	16 out of 9312	16 out of 9312	35 out of 9312
Number of 4 input LUTs:	15 out of 9312	21 out of 9312	17 out of 9312
Number of IOs	20	50	10
Number of bonded IOBs:	17 out of 232	39 out of 232	10
IOB Flip Flops:	7	28	10
Number of GCLKs	1	1	1

Загальна модель керуючого пристрою і пристрою діагностування

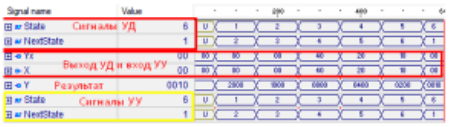
```
--Declare libraries and entity of structure model
---
--Declare components
---
--Entry to component binding
signal Yx:STD_LOGIC_VECTOR(0 to 7);
begin

      UUT0:GAS_Diagnostic_Device
           port
map(clk=>clk,reset=>reset,Yx=>Yx);

      UUT2:GAS_FSM
           port
map(clk=>clk,reset=>reset,X=>Yx,Y=>Y);
end Workstation;
```

Опис структурної моделі КП та ПД

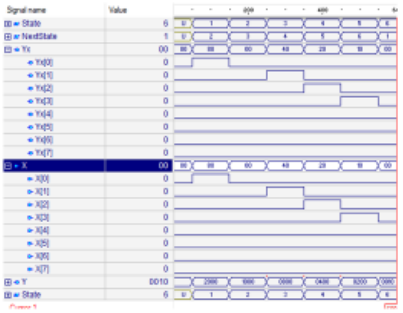


Повна Waveform з реалізацією алгоритму діагностування

Результат моделювання та синтезу роботи пристрою діагностування



Варіант обходу послідовності
a1 – a2 – a3 – a4 – a5 – a6 – a1



Selected Device : 3s500efg320-5	Варіант 1
Number of Slices	34 out of 4656
Number of Slice Flip Flops	44 out of 9312
Number of 4 input LUTs:	38 out of 9312
Number of IOs	16
Number of bonded IOBs:	17 out of 232
Number used as Shift registers: 1	1
Number of GCLKs	1

Висновки

- Дана робота демонструє можливості застосування сучасних САПР ПЛІС при проектуванні цифрових систем управління і діагностування в електроенергетиці і газопостачанні.
- Аналіз схемної реалізації (синтезу) HDL-моделей пристрою керування (керуючого автомата – КА) і апаратних засобів діагностування (діагностичного автомата – пристрій діагностування – ПД) дозволяє оцінити величину додаткових витрат апаратури за допомогою критерію Квайна.
- Аналіз додаткових витрат апаратури показав, що їх можна порівняти з апаратурними витратами на сам пристрій управління. З одного боку, це збільшує апаратурні витрати на цифрову систему керування в цілому, але, з іншого боку, при реалізації системи керування на ПЛІС, на кристалі, як правило, присутня незадіяна програмована логіка, яка може використовуватися під апаратуру діагностування.
- Якщо казати про зниження додаткових витрат апаратури на пристрій діагностування, то при автоматизованому проектуванні цифрових систем управління слід застосовувати методи тестопридатного проектування та вбудованого діагностування.

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮІК.46741Х.039ПЗ	Пояснювальна записка	60 с.
		Графічні документи	
2		Слайд-презентація	17 слайдів
		Носії інформації	
3		Компакт-диск (CD)	1 шт.

Змін.	Арк	Номер докум.	Підп.	Дата	ГЮІК.46741Х.039 ВД			
Розроб.		Гога М.В.			Вбудований пристрій діагностування кінцевих автоматів Відомість атестаційної роботи	Літ	Аркуш	Аркушів
Перевір.		Шкіль О.С.				У		
Н.контр.		Філіппенко І.В.				ХНУРЕ Кафедра АПОТ		
Затв.		Чумаченко С.В.						